NASA Technical Memorandum 89096

Introduction to the Computational Structural Mechanics Testbed

C.G. Lotts, W.H. Greene, S.L. McCleary, N.F. Knight, Jr., S.S. Paulson, and R.E. Gillian

September 1987

National Aeronautics and Space Administration

Langley Research Center
Hampton, Virginia 23665

Introduction to the Computational Structural Mechanics Testbed

Table of Contents

INTRODUCTION TO THE COMPUTATIONAL STRUCTURAL MECHANICS TESTBED

1. Summary

The Computational Structural Mechanics testbed development was motivated by requirements for a highly modular and flexible structural analysis system to use as a tool for research in computational methods and for exploration of new multiprocessor and vector computer hardware. The new structural analysis system, based on the original SPAR finite element code and the NICE System, is described. The system is denoted the CSM testbed. NICE was developed at Lockheed Palo Alto Research Laboratory and contains data management utilities, a command language interpreter, and a command language definition for integrating engineering computational modules. SPAR is a system of programs used for finite element structural analysis developed for NASA by Lockheed and Engineering Information Systems, Inc. It includes many complementary structural and thermal analysis and utility functions which communicate through a common database. Analysis examples are presented which demonstrate the benefits gained from a combination of the NICE command language with the SPAR computational modules. Testbed development to date has been carried out on a DEC VAX/VMS minicomputer; the current version is also operational on a DEC MicroVAX running ULTRIX and on a CRAY-2 running UNICOS. Future development will be directed toward UNIX systems running on multiprocessor and vector computers.

2. Introduction

Research in computational methods for structural analysis has been severely hampered by the complexity and cost of the software development process, even on traditional single processor computers. Although the researcher is usually interested in only a small aspect of the overall analysis problem, he is often forced to construct much of the supporting software himself. This time-consuming and expensive approach is frequently required because existing software that the researcher could potentially exploit is not documented in sufficient detail internally, may not be suitable because of software architecture design, or both. After enduring this time-consuming software development effort, the researcher may find that a thorough, complete evaluation of his new method is still impossible due to limitations of the supporting software. This is true, for example, in many "research-oriented" finite element codes which have a limited element library or have arbitrary restrictions on how elements of different types can be combined in a single model.

In addition, new computer architectures with vector and multi-processor capabilities are being manufactured for increased computational speed. Analysis and computational algorithms that can exploit these new computer architectures need to be developed. For credibility, these new algorithms should be developed and evaluated in standard, general-purpose finite element software rather than in isolated research software.

To address the above difficulties in computational methods research, the CSM Group at NASA Langley Research Center has undertaken the construction of a structural analysis software "testbed". The testbed provides a system which can be easily modified and extended by researchers. This is being achieved, in part, by exploiting advances in computational software design such as command languages and data management techniques.

The testbed will be used by a large group of researchers who need unrestricted access to all parts of the code, including the data manager and command language. Research on these elements of software design is needed because deficiencies in the data management strategy can have a devastating impact on the performance of a large structural analysis code, totally masking the relative merits of competing computational techniques. Furthermore, software designs that exploit multiprocessor computers must be developed. To remove all questions regarding access and use, it was decided that the testbed should be public domain software.

The decision to couple NICE and SPAR for the CSM testbed was based on four considerations. First, the details of the data management system in the original SPAR and NICE are quite similar. The data manager requests within SPAR processors are compatible with the NICE entry points. Second, the reliability, utility, and performance of the SPAR processors have been proven by almost a decade of use. Third, the concept of a high-level command language controlling the execution of independent computational modules appears to be an excellent architecture for structural analysis software. Fourth, both NICE and SPAR are public domain software.

3. NICE

3.1 Overview of NICE

The NICE (Network of Interactive Computational Elements) system (refs. 1, 2 and 3) developed at Lockheed Palo Alto Research Laboratories is an example of a modern software architecture for supporting engineering analyses. The NICE system consists of three major components: a data manager (GAL), a control language (CLAMP) for controlling analysis flow, and a command interpreter (CLIP) for interpreting CLAMP directives and decoding processor commands. Computational modules in the NICE system, called processors, are independent programs which perform a specific, well-defined task. To enforce modularity, processors do not communicate explicitly with each other but instead communicate only by exchanging named data objects in the data base. To utilize these independent processors in a particular, complex analysis task, CLAMP procedures are written to describe the analysis to be performed and the algorithm to be used. Processors access the NICE utilities by calling entry points provided in the NICE object library, implemented as Fortran 77 functions and subroutines. The set of entry points constitutes the CLIP-Processor Interface.

The NICE control language is a generic language designed to support the NICE system and to offer program developers the means for building problem-oriented languages. It may be viewed as a stream of free-field command records read from command sources (the user terminal, actual files, or processor messages). The source commands are interpreted by a "filter" utility called CLIP, whose function is to produce object records for consumption by its user program. The standard operating mode of CLIP is the processor-command mode. Commands are directly supplied by the user, retrieved from ordinary card-image files, or extracted from the global database, and submitted to the running processor. Special commands, called directives, are processed directly by CLIP; the processor is "out of the loop". Transition from processor-command to directive mode is automatic. Once the directive is processed, CLIP returns to processor-command mode. Directives are used to dynamically change run-environment parameters, to process advanced language constructs such as macrosymbols and command procedures, to implement branching and cycling, and to request services of the data manager. CLIP can be used in this way to provide data to a processor as well as to control the logic flow of the program through a single input stream. All CLIP directives are available to any processor that uses the CLIP-Processor interface entry points.

The NICE data management system is accessible to the user directly through the CLIP directives and to running processors through the GAL-Processor interface. The global database administered by the NICE-DMS is constituted by sets of data libraries (GALs) residing on direct-access disk files. Data libraries are collections of named datasets, which are collections of dataset records. The data library format supported by NICE is called GAL/82, which can contain nominal datasets made up of named records. Some of the advantages to using this form of data library are: 1) the order in which records are defined is irrelevant, 2) the data contained in the records may be accessed from the

command level, and 3) the record datatype is maintained by the manager; this simplifies context-directed display operations and automatic type conversion.

3.2 NICE Directives

Directives are special commands that are understood and processed by CLIP and not transported to the processor. A directive is to CLIP like ordinary input is to the processor. A directive is distinguished from ordinary input by beginning with a keyword prefixed by an asterisk. The keyword (directive verb) may be followed by a verb modifier, qualifiers, and parameters, as required by the syntax of the specific directive. See Reference 2, vols. 1 and 2, for a complete description of the CLAMP language. An interactive HELP facility, accessed by the *HELP directive, is built in to NICE to explain CLIP directives.

This section presents a summary of the most useful NICE directives, grouped according to their function in the NICE execution environment.

Global Data Manager Interface

*OPEN	Open data library
*CLOSE	Close data library
*TOC	Print table of contents of library
*PRINT	Print table of contents, dataset record contents, or record access table of dataset
*COPY	Copy datasets or dataset records
*DELETE	Delete dataset or record
*FIND	Returns information on libraries, datasets, or records
*RENAME	Renames dataset or record

Command Procedure Management

*SET PLIB	Set procedure library for residence of command procedures
*PROCEDURE	Initiates definition of command procedure
*CALL	Redirects input to a callable procedure ("calls" a procedure with optional argument replacement)

Nonsequential Command Processing

*IF	Conditional branching construct
*ELSE	
*ELSEIF	
*ENDIF	
*DO	Looping construct
*ENDDO	
*WHILE	While-looping construct
*ENDWHILE	

*JUMP	Transfer control to specified label
*RETURN	Force exit from command procedure
*END	Terminate definition of command procedure
	Macrosymbol Directives
*DEFINE	Define a macrosymbol or macrosymbol array
*UNDEFINE	Delete macrosymbol(s)
*SHOW MACRO	Show macrosymbols
Built-in macrosymbols	Common constants, mathematical functions, generic functions, reserved variables, boolean functions, logical functions, string catenator, string matchers, and status macros
	SuperClip Directives
*RUN	Start execution of another program
*STOP	Stops RUN-initiated execution and restarts the parent processor
	Workpool Directives
*WALLOCATE	Allocate group in the workpool
*WGET	Read database record into workpool group
*WDEF	Define macrosymbol(s) from workpool group items
*WSET	Set workpool group items to specified values
*WPUT	Write workpool group to a nominal record
	General Directives
*HELP	Lists information from NICE HELP file
*SET	Sets specified NICE control parameters
*SHOW	Shows specified NICE control parameters
*ADD	Redirects input to a text file
*DUMP	Dumps contents of any file
*REMARK	Print remark line
*UNLOAD	Unload contents of GAL library to an ASCII file
*LOAD	Load contents of GAL library from an ASCII file

3.3 NICE CLIP/GAL-Processor Interface

An application program (processor) accesses NICE utilities for command loading and data management functions by calling entry points provided in the NICE object library. The entry points are implemented as Fortran 77 subroutines or functions.

3.3.1 Clip-Processor Entry Points

The most important entry points which control command-loading actions are summarized here. See Reference 2, vol. 3 for a complete description of the usage of the CLIP entry points.

CLIP Control

CLGET	Get next command image
CLREAD	Get and parse next command
CLPUT	Insert immediate one-line message

NICE also provides many other entry points to enable the running processor to search the current command for keywords and qualifiers, to retrieve item and run information, and to evaluate macrosymbols and expressions. This is the mechanism provided to enable the processor developer to build his own problem-oriented language.

3.3.2 GAL-Processor Entry Points

The most useful entry points provided for accessing NICE GAL/82 formatted libraries are summarized here. See Reference 3 for a complete description of the GAL entry point usage.

Library File Operations

GMOPEN,LMOPEN	Open library
GMCLOS	Close library

Nominal Dataset Operations

GMPUNT,LMPUNT	Put name in Table of Contents
GMGENT	Get name from Table of Contents
GMFIND,LMFIND	Find occurrence
GMDENT,GMDEST	Delete
GMENAB	Enable
GMLINT,GMLIST	List Table of Contents

Named Record Operations

GMPUTC	Put record (character type)
GMPUTN	Put record (numeric type)
GMGETC	Get record (character type)
GMGETN	Get record (numeric type)

Supplemental Operations

GMBUDN	Break up dataset name
GMCODN	Construct dataset name
GMCORN	Construct record name
GMSIGN	Enter processor signature

3.4 Creating and Using NICE Procedures

The command procedure capability of CLIP allows the insertion of a set of predefined command records at any point in the command source stream. Selected portions of the inserted commands can be replaced by text specified in the procedure reference or "call". Commands need not be processed sequentially; branching and looping constructions may be implemented via the DO, IF and WHILE directives. CLIP procedures bear some similarities to Fortran subroutines; however, the source text is interpreted, rather than compiled, by CLIP. The procedure definition is presented to CLIP, typically by ADDing the source file that contains it. CLIP interprets the source and puts out a callable version into an ordinary data file or a data library. This version can be invoked by a CALL directive referring to the procedure name and including actual arguments to replace the formal arguments in the procedure definition.

3.4.1. Creating a procedure

a. Edit a file containing the source for one or more procedures.

b. Execute NICESPAR and "pre-process" the procedure(s) as follows:

Use the *SET PLIB directive if you want the compiled procedures to be stored in a GAL library. By default, PLIB is zero, so they are stored in an ordinary direct-access file. (This option should be used if you will be using "NICE/SPAR external processors" or invoking processors via the *RUN directive.)

Use the *ADD directive to make NICESPAR read the procedures from the source file, process them, and store them according to the PLIB setting.

3.4.2. Using a procedure

Use the *SET PLIB directive to specify where the "callable" procedures reside. Then use the *CALL directive to invoke the procedure with argument substitution as required. The *CALL directive may be used in the primary input stream or in another NICE procedure.

3.4.3. Example of creation and use of a procedure

The source code for a procedure named NLSTATIC1 resides on a file named NLSTATIC1.CLP.

"Pre-process" the NLSTATIC1 procedure to a direct access file named NLSTATIC1.DAT:

```
$ NICESPAR                ! Run NICE/SPAR (VAX/VMS)
*add NLSTATIC1.CLP        . read input from file
*eof                      . terminate execution
```

Execute the NLSTATIC1 procedure in the NICE/SPAR environment:

```
$ NICESPAR
*open 1 TEST1.LO1
*set echo=off
*call NLSTATIC1 ( Database    = TEST1.LO1       ; --
                  beg_step    = 1               ; --
                  max_steps   = 10              ; --
                  max_iters   = 7               ; --
                  beg_load    = .01             ; --
                  max_load    = 1.00            ; --
                  Nominal_DB  = TEST1.RESULTS   ; --
                  Nominal_DS  = RESPONSE.HISTORY )
[xqt EXIT
```

4. SPAR

4.1 Overview of SPAR

The computer program for Structural Performance Analysis and Redesign (SPAR) (Ref. 4) was developed in the 1970's by Lockheed Missiles and Space Company and by Engineering Information Systems, Incorporated. SPAR had its genesis in analysis technology described in Ref. 5. Early structural analysis programs which incorporated that technology included FRAME66 (circa 1966) and SNAP (circa 1970). In 1973, the concept of independent processors communicating through a global data base was demonstrated in the SPAR structural analysis system. Thermal analysis capabilities were added to the system in 1979 (Ref. 6).

The SPAR system was developed to perform stress, buckling, vibration, and thermal analysis on linear structural systems using the finite element approach. SPAR computes static deflections and stresses, natural vibration frequencies and modes, and buckling loads and mode shapes of linear finite element structural models. The structural models are composed of finite elements connected at specified joints, which can have three translational and three rotational components of deflection. Finite elements which are currently available for simulating the stiffness characteristics of a structure include axial bars, beams of general cross section, triangular and quadrilateral plates having an option to specify coupled or uncoupled membrane and bending stiffness, quadrilateral shear panels, and 4-, 6-, and 8-node solid elements. The element formulation is based on the assumed-stress hybrid formulation (Pian, Ref. 7). Properties of the plates may be specified as layers in a laminate of composite materials, and there is provision for warping of the quadrilateral plate element. Mass properties of a structure are represented by structural and nonstructural masses associated with the stiffness elements and by concentrated masses at the joints. Loading data can include any or all of the following categories: point forces or moments acting at the joints, specified joint motions, inertial loading, thermal or pressure loads, and initial strains in individual elements. Linear and nonlinear steady state and transient thermal analysis may be performed with the thermal element repertoire including conduction, convection, mass-transport, integrated, and radiation elements.

4.2 SPAR Control Language and Data Management

The SPAR system command language allows the user to design execution sequences optimally suited to the requirements of each individual application. However, no looping, conditional execution, or argument replacement is allowed in the language. Each of the SPAR processors may be invoked with a single command. All data input is accepted in free-field format. The SPAR input decoder recognizes integer, floating-point, and alphanumeric data.

Individual processors within the SPAR system are able to communicate automatically through a body of information known as the data complex. The data complex contains one or more libraries, within which may reside any number of datasets produced by the SPAR processors. Through the data complex, SPAR processors are able to generate, store, locate, and access all of the information needed to perform a particular analysis. All information generated in a run may be retained in the data complex, thereby remaining available for use in future runs. This retention is accomplished automatically without complicated restart procedures and without requiring the user to be concerned with the internal structure of the data complex.

There are several standard forms of dataset structures that are used by almost all SPAR programs. Four such dataset forms are designated TABLE, SYSVEC, ELDATA, and ALPHA. TABLE is a generalized dataset form for the storage of almost any type of data. Data such as node-point position coordinates and nodal temperatures are stored in TABLE format. SYSVEC is a special case of the TABLE form. SYSVEC is used primarily to represent the displacements and rotations at all points in a structure, and the forces and moments acting on all joints. This form is also used for diagonal mass matrices. ELDATA is a data form used to represent certain categories of data bearing a one-to-one relationship with structural elements of a given type, such as element pressure or temperature loads. The ALPHA dataset form is used to store lines of alpha-numeric text, such as static load case titles.

4.3 SPAR Processors

The SPAR processors TAB, ELD, E, and EKS are used to generate, and store in the data complex, datasets that define the finite element model of the structure. TAB and ELD are used to generate the basic definition of the structure. Subprocessors within TAB translate user input data into tables of data such as joint locations, material constants, and section properties. Subprocessors in ELD translate user input data into data tables that define individual finite elements of various types. Using the data produced by TAB and ELD, processors E and EKS generate a group of datasets, collectively known as the E-state, that contain a complete description of every element in the structure including details of element geometry and intrinsic stiffness matrices. TAB contains subprocessors which generate tables of material constants, section properties, joint locations, and various other datasets comprising a substantial portion of the definition of the structure. TAB may be used to create new datasets and to update existing datasets. Models may be modified and extended without extensive re-entering of existing input data. ELD translates element definition data from user input into datasets which are usable by other SPAR processors. Elements may be defined individually, through a variety of mesh generators or with combinations of both. An element is defined by specifying the joints to which it is connected and by pointers to applicable entries in tables, such as section properties and material constants. As the ELD input is processed, extensive error checks are performed. The E processor constructs, in skeletal form, an "element information packet" for each element in the structure. The E processor supplies this packet with general information

such as connected joint numbers and table reference numbers, material constants, geometrical data, and section property data. In addition, the system mass matrix, which is in lumped-mass diagonal form, is generated. The EKS processor completes the "element information packets" by computing and inserting intrinsic stiffness and stress matrices.

The six processors TOPO, K, M, KG, INV, and PS are directly associated with the assembly, factoring, and display of SPAR-format system matrices. SPAR uses a procedure for solving high-order systems of linear equations of the kind which occur in displacement method finite element analysis. The system stiffness matrix is regarded as an array of sub-matrices; each sub-matrix is of size n-by-n, where n is the number of degrees of freedom at each joint. The non-zero sub-matrices in a system stiffness are those corresponding to pairs of joints connected by elements. Accordingly, in all but the smallest finite element models, only a small fraction of the sub-matrices are non-zero. The characterizing feature of the SPAR procedure is that it operates exclusively with data contained in the non-zero sub-matrices; this virtually eliminates the unessential arithmetic and wasted storage space associated with conventional band matrix techniques.

Processor TOPO analyzes element interconnection topology and creates the datasets KMAP and AMAP. KMAP is used by processors K, M, and KG to guide assembly of system stiffness and mass matrices in the SPAR standard sparse-matrix format. AMAP is used by processor INV in factoring system matrices. The K processor assembles unconstrained system stiffness matrices in the sparse-matrix format. The M processor assembles unconstrained system consistent mass matrices using only the structural and nonstructural distributed mass associated with the elements. The KG processor forms and assembles unconstrained system initial stress (or geometric) stiffness matrices based on the stress state currently embedded in the E-state dataset. The INV processor forms and factors assembled system matrices in the sparse-matrix format subject to specific constraint sets which have been defined in the processor TAB. SPAR-format matrices and factored SPAR-format matrices may be printed with the PS processor.

The SPAR processors AUS, DCU, and VPRT are general utility programs for use in execution of the SPAR system. The AUS processor is an arithmetic utility program performing an array of functions in the areas of: matrix arithmetic, construction, editing, and modification of data tables. The DCU processor is a set of utility functions for management of the data complex. The VPRT processor is used to display any dataset in the SYSVEC format (e.g., eigenvectors, static displacements, reactions, and nodal load vectors).

Basic analysis computational functions are performed in processors SSOL, GSF, PSF, EIG, and DR. SSOL computes displacements and reactions due to a given set of nodal loads (e.g., point loads, equivalent nodal loads due to pressure). GSF generates datasets containing element stress and internal load information. The PSF processor is used to print element stresses and internal load data in GSF created datasets. The EIG processor solves linear vibration and bifurcation buckling eigenproblems. EIG implements an iterative process consisting of a Stodola (matrix iteration) procedure followed by a Rayleigh-Ritz procedure. This process is used repetitively resulting in successively refined approximations of the eigenvectors associated with a specified number of eigenvalues. DR computes the transient response of an uncoupled system using a matrix series expansion method.

Thermal analysis functions are performed in processors TGEO, MTP, VIEW, TRTA, TRTG, TAFP, TRTB, SSTA, TADS and TAK. TGEO computes element local coordinates and performs element geometry checks. MTP computes fluid mass-transport rates. VIEW computes radiation view factors. TRTA, TRTB, and TRTG generate transient solutions by explicit, implicit, and GEAR methods, respectively. TAFP computes element and nodal heat rates. SSTA generates linear or nonlinear steady state solutions. TADS and TAK are utility processors for debugging and dataset format conversion.

5. The CSM Testbed

5.1 Integration of NICE and SPAR

NICE/SPAR is structured as a NICE "macroprocessor" with a central executive module which calls the installed SPAR processors as subroutines. The SPAR processors are data-coupled through the common global database, each interfacing with CLIP through the common input routines and with GAL through a small set of database interface routines. The macroprocessor configuration was selected for efficiency considerations related to opening and closing data base files. User-developed processors may be implemented as "external processors", which are independent executable programs that may be invoked from the NICE/SPAR command stream.

Because the data management philosophy of NICE is similar to the SPAR approach, the installation of the SPAR computational processors under NICE was relatively straightforward. Usage descriptions of the primary SPAR processor/NICE data management interface subroutines are given in Appendix E. These routines (DAL, RIO, TOCO and LTOC) are used by the existing NICE/SPAR processors as the bridge between the SPAR data management method and the NICE nominal dataset/named data record utilities.

NICE/SPAR data libraries are written to disk files named (by default) NS.Lxx, where xx is the library number (01-30). Most processors use library number 1. However, by using the CLAMP *OPEN directive, a user can explicitly associate any legal external file name with a library. The data libraries are in NICE/GAL82 format; datasets are nominal datasets using the same naming convention as SPAR. Records of the datasets are named records (currently implemented with all records simply named DATA); datasets are written as one record per SPAR block, or one record per SPAR data segment. The current NICE/SPAR dataset contents are described in Appendix A.

To implement the interface with the NICE command language interpreter, the SPAR input processor (READ) was modified to use the CLIP routine CLGET; command input received from CLIP is parsed in READER according to the SPAR syntax described in Reference 4, with minor modifications noted in Appendix D of this report. The SPAR termination routine (FIN) was modified to use the CLIP routine CLPUT to send a message to CLIP to terminate execution. The usage descriptions for these routines are described in Appendix E.

5.2 Installing and Running NICE/SPAR

5.2.1 NICE/SPAR installation on VMS

NICE/SPAR is currently implemented on a VAX minicomputer running under the VMS operating system. A delivery tape is available for installing the software on another VMS machine; see Appendix B for complete instructions for performing the installation.

The delivery tape contains the source code for both NICE and the testbed processors. It also contains object libraries, procedures for linking user-created processors, demonstration problem procedures, and the executable version of the testbed program.

Once the installation is complete and the symbols and logical names are properly set up, the testbed program is ready to be executed.

5.2.2 NICE/SPAR Usage on VMS

The NICE/SPAR executive is invoked by typing NICESPAR in the interactive mode. The command used to invoke a NICE/SPAR processor is "[XQT processor-name"; the command to exit a processor and the NICE/SPAR executive is "[XQT EXIT". NICE directives (prefixed by *) may be entered, intermixed with SPAR commands. Batch mode processing is also available, with all commands and directives supplied from a disk file.

Because NICE converts all input (except labels) to uppercase characters, which SPAR requires, raw input data may be entered in either upper or lower case.

NICE directives are documented in Reference 2. SPAR commands are documented in Reference 4. Differences between NICE/SPAR and the documented version of SPAR are described in Appendix D.

5.3 Running NICE/SPAR on the NAS CRAY-2

The NICE/SPAR software is operational on the NAS CRAY-2 computer under the UNICOS operating system. To access this version of the program, the name of the directory where the executable file resides should be inserted in the user's PATH environment variable; then to execute the program, type "nicespar."

VAX/VMS command procedures for running NICE/SPAR may be converted into UNICOS script files by 1) deleting all DCL commands and replacing with corresponding UNICOS commands, if necessary; 2) replacing the command line which initiates NICESPAR execution with one that contains "time nicespar << \eof"; and 3) adding a line at the end which contains "\eof". The UNICOS script file should be made executable with the chmod command. The script may then be executed by entering its file name.

The primary difference in usage between the VAX/VMS version and the UNICOS version of NICE/SPAR is that the case of text entered for names (procedure, file, dataset, record, etc.) in NICE directives is retained in the UNIX version, instead of automatically being converted to uppercase as in the VMS version. However, text interpreted by the SPAR READER subroutine (used by the SPAR analysis processors) is converted to uppercase before interpreting in both versions. Another difference in usage is that file names may be specified only with reference to the current directory (full pathnames may not be used).

5.4 NICE/SPAR Installed Analysis Modules

The following is a list of SPAR processors and newly developed CSM processors (indicated by +) which are currently installed in NICE/SPAR. The CSM processors are described in Appendix C.

	AUS	- Arithmetic Utility System
+	CSM1	- Focus Problem Mesh Generation (See Appendix C.)
	DCU	- Data Complex Utility
	DR	- Linear Dynamic Response Analyzer
	E	- E-State Initiation
	EIG	- Sparse Matrix Eigensolver
	EKS	- Element Intrinsic Stiffness and Stress Matrix Generator
	ELD	- Element Definition Processor
	EQNF	- Equivalent Nodal Force Generator
+	ENL	- Element Nonlinearity Processor
	GSF	- Stress Data Generator
	INV	- SPAR Format Matrix Decomposition Processor
	K	- System Stiffness Matrix Assembler
	KG	- System Initial Stress (Geometric) Stiffness Matrix Assembler
+	LAU	- Laminate Analysis Utility (See Appendix C.)
	M	- System Consistent Mass Matrix Assembler
	MTP	- Fluid Network Analyzer
	PAMA	- AMAP Dataset Printer
	PKMA	- KMAP Dataset Printer
	PS	- SPAR Format Matrix Printer
	PLTA	- Plot Specification Generator
	PLTB	- Production of Graphical Displays
	PRTE	- EFIL Dataset printer
	PSF	- Stress Table Printer
+	RSEQ	- Renumbering Strategies (See Appendix C.)
	SSOL	- Static Solution Generator
	SSTA	- Steady State Thermal Analyzer
	TAB	- Basic Table Input
	TADS	- Thermal Analysis Debugging Utility
	TAFP	- Flux and Heat Rate Processor
	TAK	- Skyline K Matrix Converter
	TGEO	- Thermal Element Geometry Processor
	TOPO	- Element Topology Analyzer
	TRTA	- Explicit Transient Thermal Analyzer
	TRTB	- Implicit Transient Thermal Analyzer
	TRTG	- Gear Method Transient Thermal Analyzer
+	VEC	- Vector Algebra Processor
	VIEW	- Radiation View Factor Processor
	VPRT	- Vector Printer

5.5 Structural Analysis Example Problems

During NICE/SPAR development, many analysis problems have been designed and executed to verify the correctness of the system. Four of these problems are presented here to illustrate the input syntax, analysis flow, and use of typical CLAMP directives in describing analysis algorithms. The source for VAX/VMS DCL procedures for these examples is provided on the installation tape; refer to the `[NICESPAR.DEMO]` directory.

The first problem is the static stress analysis of a section of a toroidal shell. The input for this example is shown in figure 1. The shell wall consists of four layers of composite material with orientations $90°/0°/\pm 45°$. The finite element model consists of 337 nodes and 320 combined membrane-bending elements. (The SPAR designation for this element type is E43). This example demonstrates the relatively straightforward usage of NICE/SPAR for a small, sequential analysis problem. Processors TAB and ELD are used to input all geometrical and property data describing the model. The JREF command in TAB is used to align the joint reference frames with the shell coordinate system. Both the applied loading (defined in AUS) and the calculated displacements (from processor SSOL) are relative to these reference frames. Later in the analysis, the calculated displacements and reactions are converted to the global reference frame using the LTOG (local-to-global) command in AUS and then printed using processor VPRT. Stress information is calculated by processor GSF and then selectively printed in three different formats by separate executions of processor PSF.

The second example is the dynamic analysis of a planar, cantilever beam. The analysis is carried out using both a modal method and a direct integration of the system equations of motion using the Newmark integrator. This example shows how the SPAR processors and the NICE CLAMP command language can work together to perform a fairly complex analysis task.

The input for this example is shown in figure 2 and consists of five NICE CLAMP procedures. Procedure CANT_BEAM defines the beam model and calculates system stiffness and consistent mass matrices. The beam is excited by an initial displacement which is the static deformation shape resulting from a unit applied displacement at the tip.

If a modal transient response is being performed, procedure VIBR_MODES is called, followed by procedure TR_MODAL. A formal argument, nmodes, in VIBR_MODES indicates the number of vibration modes to be calculated. A similar parameter in TR_MODAL indicates the number of modes to be used in the transient response analysis. SPAR processor DR integrates the modal equations and performs the back transformation for selected physical coordinates.

If a transient response calculation by direct integration of the system equations is being performed, procedure TR_DIRECT is called, which in turn calls procedure NEWMARK. Procedure NEWMARK implements the well known Newmark integration method for second-order, coupled systems. Parameters such as system stiffness and mass matrix names, the time step, and the total number of time steps in the analysis are formal arguments to procedure NEWMARK. In NEWMARK, extensive use is made of the CLAMP macro expression capability for calculating integration constants and controlling the algorithm. The initial acceleration at time t = 0 is calculated from the given initial displacement and velocity vectors. This is done by using processor AUS to set up the equations of motion at t = 0, and INV and SSOL to solve for the acceleration. At each subsequent time step, processor AUS is used to set up the recursion relations, and processor SSOL is used to solve for the displacement vector at the next time step. Then velocity and acceleration vectors can be calculated and selectively printed. Although procedure NEWMARK is not intended as a "production" quality implementation of the Newmark method, it does illustrate many of the features and the potential of NICE/SPAR procedures in facilitating methods research and development.

The third problem is the determination of the buckling load of a blade-stiffened composite panel, with a central hole and discontinuous stiffener, subjected to a uniform end shortening (the CSM Focus Problem 1). The input for this example is shown in Figure 3. The finite element model consists of 388 nodes and 344 combined membrane-bending (SPAR E43) elements.

This example uses four procedures and two CSM developed processors to determine the buckling load of the panel. The procedure MESH_FOCUS provides input data required by the processor CSM1; procedure MATDAT provides the input data required by processor LAU. The values entered in the TABLEs formed in MESH_FOCUS define the finite element model. By changing TABLE values, one may refine the mesh, change boundary conditions, grade the mesh around the hole, fill in the hole, etc. The TABLEs formed in MATDAT define material and section properties.

The processor CSM1 actually generates the finite element model of the Focus Problem. Using the data supplied in MESH_FOCUS, CSM1 generates nodal coordinates, element connectivities, applied displacements, and boundary conditions in the form of a file, PANEL.PRC, written to the user's default directory. PANEL.PRC, containing the problem data in the form of CLAMP procedures, is then incorporated into the runstream using the CLAMP directive *ADD. The procedures included in PANEL.PRC may then simply be called when needed.

Procedure MAIN acts as a driver for the solution of the buckling problem. Since this problem may be solved using either E43, E33, or experimental elements, the CLAMP *BLOCK IF directive is used to avoid execution of SPAR processors which do not recognize the experimental elements. The final procedure, PLOT_SPEC, sets up plot specifications for the model. Included at the end of the figure is a Table of Contents of Library 1, listing all existing datasets and their sizes (in number of records) at the end of execution of the example procedure.

The fourth example problem is also a buckling problem. A square plate is subjected to a uniform end shortening. Only one quarter of the plate has been modeled and symmetry boundary conditions have been imposed at the x = 0. and y = 0. edges. This runstream was used to evaluate the sensitivity of both SPAR and experimental elements to mesh distortion and refinement.

The finite element model is shown in Figure 4. There are 7 joints per side for a total of 49 joints. Either 4- or 9-node elements may be used. If 4-node elements are being examined, then there are 36 elements total with a block of 4 elements in the center rotating rigidly through the skew angle, Θ. For 9-node elements, there are 9 elements with one central element rotating rigidly through the angle Θ.

The input runstream, containing four procedures, is shown in Figure 5. At the top of the runstream, several global macros, which give the user control of the model, have been defined. The driving procedure, SKEW_GRID, makes extensive use of CLAMP macros for defining joint locations. The user has only to specify the skew angle and the nodal coordinates will be calculated accordingly.

Procedure MATDAT is used to set up the tables of material and section properties required by processor LAU. Procedures G41 and G91 are mesh generators for a uniform grid for the experimental 4- and 9-node elements, respectively. The Table of Contents for Library 1, listing all of the datasets existing at the end of execution, is given at the end of the figure.

6. New Processor Development and Database Interface

A major goal of the NICE/SPAR system is to provide mechanisms for easy interface of user-supplied computational processors. Existing processors can be modified or new processors can be added to the testbed. The most critical issue faced by the developer or modifier of a processor is maintaining compatibility with existing processors.

Compatibility with the system is insured by making the input and output data structures (datasets) compatible with those of other processors and making the capabilities of the new or replacement processor complement those of existing processors. The internal structure of many NICE/SPAR datasets is described in Appendix A. If the intent of a processor developer is to replace completely an existing processor, then typically both the input and output data sets and capabilities would at least be equal to those of the replaced processor. Processors developed to merely augment existing processors do not have so rigorous a requirement. As long as their input and output datasets agree with those of processors with which they interact, compatibility is assured.

Sometimes this compatibility is easy to achieve and sometimes it is more difficult. Two examples serve to illustrate these extremes. The first example is the addition of a new, special-purpose processor to do interactive plotting of the geometry of a finite element model. It would be necessary for this processor to read the datasets containing joint locations and connectivity information for different elements. These datasets are relatively simple to access. And since no output datasets need be produced, no compatibility problem is introduced. A more difficult processor development task would be the replacement of the existing system matrix factorization processor, INV. SPAR uses a storage scheme for system matrices that involves storing only the non-zero blocks in the upper half of the (assumed) symmetric matrix. Processor TOPO performs the complicated task of determining the necessary information required for assembly and factorization of these matrices. A capability compatible with TOPO would not be easy to produce. Finally, the dataset output by INV has a special form and several processors in the system require this particular form. As a result, development of a new INV with identical input and output datasets would require careful study. The alternative of replacement of the basic system matrix data structures would have an impact on many processors in the code. Consequently the effect of this type of change on computational efficiency, generality, and extendability would have to be carefully considered.

6.1 Techniques for Interfacing with the NICE/SPAR Database

The NICE/SPAR user has several options for accessing and manipulating data which currently exists in a NICE/SPAR global database:

a. The user may extract the data in text form from the database to a disk file to be processed as formatted data. The processors which exist for formatting some of the more complex datasets are PRTE, PS, PAMA, and PKMA. The CLAMP *PRINT directive also allows the user to extract the contents of named records in fixed format fields. With either of these options the output can be redirected to a disk file via the CLAMP *SET directive.

b. The user may use the AUS processor to perform a variety of arithmetic and dataset construction operations.

c. The user may write a new program or modify an existing one to directly access the database via the GAL-Processor interface routines. Guidelines for writing and integrating processors are given in the next section.

6.2 Guidelines for New Processor Development

There are two ways to integrate new processors into the NICE/SPAR environment; the developer can select the method which is more appropriate to his application. The first method is to build the processor as a NICE/SPAR "external processor", which is an independent executable program that may be invoked directly from a NICE/SPAR input stream. The second method is to install the processor into NICE/SPAR directly and create a single executable program that contains the NICE/SPAR processors and the newly developed processor.

The advantages of the first method are that a) linking the processor takes a shorter time and that b) the resulting executable file is smaller. The advantages of the second method are that a) runtimes for installed processors are shorter than for external processors because library files are not closed and reopened; b) NICE procedures invoking the processor may be resident in GAL libraries; and c) the developer can customize his NICE/SPAR executive to include only the modules which his application requires. The former method is recommended for initial checkout of a processor, while the latter is recommended for processors which will be executed regularly.

The processor developer should follow the guidelines given below for coding and integrating his new processor.

6.2.1 Guidelines for Coding New Processors

a. The name of the processor should be no longer than 4 characters; this should be the name of the source file with the extension ".FOR." This name must not be one of the installed processor names listed in section 5.3.

b. The processor should be written in standard Fortran 77 language in the form of a subroutine whose name is the processor name. The subroutine should have no arguments.

c. The processor should begin execution with a call to the library subroutine INTRO with the processor name as the only argument. The given name is used by the GAL data manager as the creating processor for new datasets inserted in GAL libraries; it also appears in the interactive prompt string if the SPAR READER routine is used for input command processing. See Appendix E for the usage description of INTRO.

d. The processor should perform the following steps before terminating or returning to the calling program: (1) Call library subroutine NSNEXT with the only argument being the name of the next processor to be executed. This step is not required if the SPAR READER routine is used for input command processing. (2) Close any files (other than GAL library files) opened during this processor's execution. (3) Call library subroutine FIN to close GAL libraries. See Appendix E for usage descriptions of NSNEXT and FIN.

e. The labeled common block /IANDO/ with 2 integer variables containing user input and output unit numbers should be included in any subroutine which uses FORTRAN write statements. The unit numbers are assigned in the subroutine INTRO. All write statements should refer to the above output unit. This unit number is assigned to correspond to the NICE PRT unit at the beginning of processor execution.

f. The NICE CLAMP utilities (Ref. 2) or the NICE/SPAR subroutine READER (Appendix E) should be used to process command input; the READER routine uses the NICE CLGET routine to filter NICE commands and interprets user input according to the SPAR command format.

g. The NICE/SPAR data handling utilities DAL and RIO (Appendix E) may be used to interact with the database. However, new processor developers may use the NICE data manager calls (Ref. 3) directly.

h. If the processor is to be executed as an independent program or as a NICE/SPAR "external processor", include a main program which calls the processor subroutine.

i. Logical unit numbers 1 through 40 should not be used for files other than libraries to avoid possible conflicts with CLIP and GAL.

6.3 NICE/SPAR Processor Integration on a VAX/VMS System

Depending on the integration method chosen above (external or installed processor), the processor code should be linked with libraries using VMS procedures provided with the NICE/SPAR software as follows:

6.3.1 Integrating a NICE/SPAR External Processor

a. The default directory should be set to the directory where the processor source file resides. The processor should be compiled and linked using the command

```
@NS$SRC:BLDEXTP processor-name
```

Executing this command procedure will create a compiler listing file and an executable file in the default directory.

b. The processor can be executed in the NICE/SPAR environment as follows:

1. The default directory should be set to the one where the executable file for the experimental processor and NICE/SPAR data libraries reside.
2. Type NICESPAR to invoke the executive program.
3. Enter optional NICE/SPAR commands.
4. Enter the NICE/SPAR command "[XQT processor-name" to start execution of the processor.

6.3.2 Installing a New Processor into NICE/SPAR

The easiest way to install a new processor into NICE/SPAR is to use the name of a "dummy" processor, EXP1, EXP2, EXP3, EXP4, or EXP5, as the name of the subroutine and source file. Then compile the new processor and link it into NICE/SPAR with the following command:

```
@NS$SRC:BLDNEWNS processor-name
```

This will create an object file, a compiler listing file, and an executable file in the default directory. To execute this new version of NICE/SPAR, you must first define the logical name NS$EXE for your process to be the directory in which the new executable file resides. For example: `DEFINE NS$EXE DUA0:[SUE.SPAR]`. Then type NICESPAR to execute the program.

Another way to install a new processor in NICE/SPAR is to modify the main program for NICE/SPAR to add the name of your processor to the list of known processors and add a statement to call your subroutine; then compile the main program and your processor and link your processor with the new main program and the other NICE/SPAR processors. The specific steps to be performed are:

a. `COPY NS$SRC:NICESPAR.FOR []`

b. Edit NICESPAR.FOR to include the name of your processor and to call it as a subroutine.

c. Compile the main program and your processor, creating object files in the default directory.

d. Link your new version of NICESPAR using the command:

```
@NS$SRC:LINKNEWNS  NICESPAR processor-name
```

To execute this version of NICESPAR, you must first define the logical name NS$EXE for your process to be the directory in which the new executable file resides. For example: `DEFINE NS$EXE DUAO:[SUE.SPAR]`. Then type NICESPAR to execute the program. Use the command "`[XQT` processor-name" to run your processor.

6.4 NICE/SPAR Processor Integration on the NAS Cray-2

Procedures and "makefiles" for integrating user-written processors into a NICE/SPAR executable file are currently under development.

6.5 Installing User Elements

The mechanism provided in SPAR for installing user elements is the "experimental element" provision. The user must write subroutines to be linked with NICE/SPAR to replace "dummy" routines in the installed version of the code. The "experimental element" routines are DMEXPE, KEXPE, CMEXPE, and KGEXPE in processors E, EKS, M, and KG, respectively. The minimum requirement for incorporating experimental elements is to provide the routine KEXPE; the others may be omitted. See Appendix F for a complete description of the use of this capability, including the calling sequences and argument definitions of the user-written subroutines.

The installed version of processors EKS and KG have a family of experimental elements already incorporated, the C^0 (shear-deformable) shell elements. Processor EKS has the subroutine KEXPE installed for computing experimental element stiffness and stress recovery data, which is incorporated into the element "EFIL" dataset by the driver routines in EKS. Processor KG has the subroutine KGEXPE installed for computing experimental element initial-stress stiffness data. These two processors could be used as models for installing a different family of elements, replacing the routines KEXPE and KGEXPE with new user written routines.

After the subroutines are written and merged with the source code for the corresponding NICE/SPAR processors into source files in the user's directory, "external proccessors" may be created using the procedure described in Section 6.3.1. The source file names should be different from the original file names; for example, change EKS.FOR to EKSX.FOR. The name of the external processor to be referred to in a NICE/SPAR execution would then be EKSX.

7. CSM Testbed Research Directions

The testbed will be used to develop and evaluate new structural analysis and computational methods, carry out applications studies, and provide requirements for a new structural analysis system that exploits advanced computers. To that end, the testbed will be a modern, modular system that handles data efficiently, that contains a command language which is powerful and easy to learn and use, and that has an architecture which allows users to add and modify software with minimal difficulty. In keeping with the CSM philosophy that analysis methods are developed in the context of problem solving, the testbed's structural analysis capability will increase as additional applications studies are carried out. The current testbed has been developed on a DEC VAX/VMS minicomputer and has been installed on a DEC MicroVAX running ULTRIX as well as a CRAY-2 running UNICOS. Future development will be directed toward UNIX systems running on multiprocessors and vector computers, including supercomputers.

CSM contracts, grants, inhouse research, and interactions with the aerospace industry and other government research organizations will guide the testbed activity to meet the above objectives.

Figure 1. NICE/SPAR Input for the Composite Toroidal Shell Example

```
$!
$! NICE/SPAR DEMONSTRATION PROBLEM 13
$! COMPOSITE TOROIDAL SHELL
$!
$ SET VERIFY
$ SET DEF NICESPAR$DEMO
$ nicespar
*set echo=off
*open 1, demo13.101 /new
[XQT TAB
  ONLINE=0
  START 337
 title' composite toroidal shell
  JLOC: FORMAT=2
   2 650.0125 0. 0.       650.0125  5.2888 0.       21  16
   3 650.1866 0. -.8754   650.1866  5.2888 -.8754   21  16
   4 650.1866 0. +.8754   650.1866  5.2888 +.8754   21  16
   5 650.6825 0. -1.6175  650.6825  5.2888 -1.6175  21  16
   6 650.6825 0. +1.6175  650.6825  5.2888 +1.6175  21  16
   7 651.4246 0. -2.1134  651.4246  5.2888 -2.1134  21  16
   8 651.4246 0. +2.1134  651.4246  5.2888 +2.1134  21  16
   9 652.3    0. -2.2875  652.3     5.2888 -2.2875  21  16
  10 652.3    0. +2.2875  652.3     5.2888 +2.2875  21  16
  11 653.1754 0. -2.1134  653.1754  5.2888 -2.1134  21  16
  12 653.1754 0. +2.1134  653.1754  5.2888 +2.1134  21  16
  13 653.9175 0. -1.6175  653.9175  5.2888 -1.6175  21  16
  14 653.9175 0. +1.6175  653.9175  5.2888 +1.6175  21  16
  15 654.4134 0. -.8754   654.4134  5.2888 -.8754   21  16
  16 654.4134 0. +.8754   654.4134  5.2888 +.8754   21  16
  17 654.5875 0. 0.       654.5875  5.2888 0.       21  16
   1 652.3    5.2888  0.
  MATC: 1 .114+07 0.28
  BA:  DSY 1 .675-03 0. .675-03 0. .09 .270-02  :  .
  MREF: 1 1 2 1 .99574
  JREF:  NREF=-1: 1,337
  CON=1: FIXED PLANE=2
  SA(4)
     FORMAT=laminate: 1 .   4 LAYER COMPOSITE
        -9.375-03 90. .00625> .   LAYER 1, INSIDE SURFACE
        1.8560+05 2.0010+03 7.1470+03 0. 0. 4.0620+03>
        6.0400-01 6.5140-03 2.3260-02 0. 0. 1.3220-02
        -3.125-03 0.0 .00625> .   LAYER 2
        1.8560+05 2.0010+03 7.1470+03 0. 0. 4.0620+03>
        6.0400-01 6.5140-03 2.3260-02 0. 0. 1.3220-02
        3.125-03 45. .00625> .    LAYER 3
        1.8560+05 2.0010+03 7.1470+03 0. 0. 4.0620+03>
        6.0400-01 6.5140-03 2.3260-02 0. 0. 1.3220-02
        9.375-03 -45. .00625> .   LAYER 4, OUTSIDE SURFACE
        1.8560+05 2.0010+03 7.1470+03 0. 0. 4.0620+03>
        6.0400-01 6.5140-03 2.3260-02 0. 0. 1.3220-02
```

Fig. 1-1

```
2 .  4 LAYER COMPOSITE DIFFERENT INPUT FORMAT
-.009375 90. .00625 185600. 2001. 7147. 0. 0. 4062. .604  .0065 .023
0. 0.
.0132
-.003125 0.0 .00625 185600. 2001. 7147. 0. 0. 4062. .604  .0065 .023
0. 0.
.0132
 .003125 45. .00625 185600. 2001. 7147. 0. 0. 4062. .604  .0065 .023
0. 0.
.0132
.009375 -45. .00625 185600. 2001. 7147. 0. 0. 4062. .604  .0065 .023
0. 0.
.0132
3 .  4 LAYER COMPOSITE DIFFERENT INPUT FORMAT AND VALUES
  -9.375-3  90. .00625 1.856+5 2.001+3 7.147+3  0. 0. 4.062+3
   -3.125-3 0. .00625 1.856+5 2.001+3 7.147+3  0. 0. 4.062+3
  3.125-3 45. .00625 1.856+5 2.001+3 7.147+3 0. 0. 4.062+3
  9.375-3 -45. .00625 1.856+5 2.001+3 7.147+3 0. 0. 4.062+3
[XQT DCU .
PRINT 1 SA .
[XQT AUS
  SYSVEC: APPLIED FORCES 1
     CASE 1: I=3: J=1: 1.0
     CASE 2: I=2: J=1: 322,337: 0.058824
  ALPHA: CASE TITLE 1
     1' TRANSVERSE SHEAR LOAD
     2' AXIAL LOAD
[XQT ELD
  ONLINE=0
  E43
  GROUP 1' 0 TO 22.5 DEG.
       2  18   19 3  1 20 1
  GROUP 2' 22.5 TO 180 DEG.
       3  19   21 5  1 20 7
  GROUP 3' 180 TO 202.5 DEG.
       17 33   32 16 1 20 1
  GROUP 4' 202.5 TO 360 DEG.
       16 32   30 14 1 20 7
  E21: 1  322  3  16 1
 ONLINE=1
[XQT E
T= .1-19,-.001,.0001,.0001,20.,.0001,.0001,.0001
[XQT EKS
[XQT TOPO
[XQT K
[XQT INV
 ONLINE=2
[XQT SSOL
[XQT AUS
  DEFINE D=STAT DISP
  DEFINE R=STAT REAC
  GLOB DISP=LTOG(D)
  GLOB REAC=LTOG(R)
[XQT DCU
  TITLE 1' 337 JOINT COMPOSITE TOROIDAL SHELL
  TOC 1
[XQT GSF
  E43: 1: 3 .
[XQT PSF
[XQT PSF
  RESET DISP=2, CROS=0, NODES=0
  DIV=1. .001 .001 1.
[XQT PSF
  RESET DISP=3, CROS=0, NODES=0
```

Fig. 1-2

```
  DIV=1. .001 .001 1.
[XQT VPRT
 JOINTS=2,322,16: 9,329,16: 17,337,16: 10,330,16 .
  TPRINT STAT DISP
  TPRINT GLOB DISP
 JOINTS=2,17
  TPRINT STAT REAC
[XQT DCU
 TOC 1
[xqt exit
```

Fig. 1-3

Figure 2. Clamp Procedures for Transient Response Analysis of a Cantilever Beam

```
$ SET VERIFY
$ set def ns$demo
$ del cbeam.101;*,cbeam.102;*,ns.*;*,cbeam.128;*
$ nicespar
*set echo off
*set plib = 28
*open 28 cbeam.128
*open  1 cbeam.101

*def/i jt = 11
*procedure CANT_BEAM
[xqt tab
 start <jt> 3,4,5
 jloc
 1 0. 0. 0.     25. 0. 0. <jt>
 matc
 1 10.+6 .3 .101
 ba
    rect 1 1.0 .1
 mref
 1 1 2 1 1.0
 con 1
 zero 1,2,6 : 1
 con 2
 zero 1,2,6 : 1
 nonzero 2 : <jt>
[xqt eld
 e21
*def/i jtm1 = <<jt> - 1>
 1 2 1 <jtm1>
[xqt e
[xqt eks
[xqt topo
[xqt k
[xqt m
 reset g=386.
.
. compute initial displacement due to a static end load
.
[xqt aus
 sysvec : appl moti
 i=2 : j=<jt> : -1.0
[xqt inv
 reset con=2
[xqt ssol
 reset con=2
[xqt dcu
 change 1 stat disp 1 2 u0 aus 1 1
[xqt dcu
 toc 1
*end

*procedure VIBR_MODES (nmodes)
.
. computes "nmodes" vibration modes
.
*def/i nmodes = [nmodes]
[xqt inv
*def init = <min(<2*<nmodes>> ; <<nmodes> + 8>)>
[xqt eig
 reset init=<init>, nreq=<nmodes>, m=cem
```

Fig. 2-1

```
[xqt vprt
 vectors = 1, <nmodes>
 print vibr modes
[xqt dcu
 toc 1
*end

*procedure TR_MODAL (nmodes)
.
. performs transient response analysis (modal superposition)
.
*def/i nmodes = [nmodes]
[xqt aus
  define x = vibr mode 1 1 1,<nmodes>
  define e = vibr eval
. compute modal initial displacements
  define id = 1 u0 aus 1 1
  idm = prod(cem, id)
  iqx = xty(x,idm)
  table(nj=<nmodes>) : xtmx : j=1,<nmodes> : 1.0
  table(nj=<nmodes>) : xtkx : transfer(source=e)
  table(ni=1,nj=<nmodes>) : td
*def/i sbase = <(<jt> - 1)*3 + 1>
*show/macro sbase
  transfer(source=x, sbase=<sbase>, ilim=1)
[xqt dr
  dtex(dt=.001)
  tr1(qxlib=1,qx1lib=1,t1=0.0,t2=.12)
  back
   t = td : y = qx
   z = zd aus
[xqt dcu
  toc 1
  print 1 iqx
  print 1 td
  print 1 zd
*end

*procedure TR_DIRECT
.
. performs transient response analysis by direct integration
. of the equations of motion
.
[xqt aus
  sysvec : ud0 . initial velocities = 0
    i = 1 : j = 1 : 0.0
*open 2 cbeam.102
*call NEWMARK (mname = cem; delt = .001; nstep = 100; pfreq = 10)
*end

*procedure NEWMARK (              --
                    kname = k   ; -- . first name of global k
                    mname = dem; -- . first name of global m
                    beta  = .25; --
                    gamma = .50; --
                    delt =  0.0; -- . time step
                    nstep       ; -- . number of time steps
                    slib  =   2; -- . number for temp. library
                    pfreq =   1  -- . print frequency for results
                   )
.
. Performs dynamic analysis on a linear system using the
. Newmark-Beta implicit integration method
```

Fig. 2-2

```
.
.
. Initialization
.
*def/a kname = [kname]
*def/a mname = [mname]
*def/e beta  = [beta]
*def/e gamma = [gamma]
*def/e delt  = [delt]
*def/i nstep = [nstep]
*def/i slib  = [slib]
*if <delt> /eq 0.0 /then
   *remark error: time step (delt) = 0.0
   *stop
*endif
*def/e a0 = (1.0/<beta>/<delt>/<delt>)
*def/e a1 = (<gamma>/<beta>/<delt>)
*def/e a2 = (1.0/<beta>/<delt>)
*def/e a3 = (1.0/2.0/<beta> - 1.0)
*def/e a4 = (<gamma>/<beta> - 1.0)
*def/e a5 = (<<gamma>/<beta> - 2.0>*<delt>/2.0)
*def/e a6 = (<1.0 - <gamma>>*<delt>)
*def/e a7 = (<gamma>*<delt>)
*def/e ma2 = <-<a2>>
*def/e ma3 = <-<a3>>
*show macro
.
[xqt aus
 khat = sum(<kname>, <a0> <mname>)
. calculate initial acceleration vector
.
 inlib = 3 : outlib = 3
 define k = 1 <kname>
 define u0 = 1 u0
 appl forc 1 = prod(k, -1.0 u0)
[xqt inv
 reset k = <mname>, kilib=3, dzero = 1.e-9
[xqt ssol
 reset k=<mname>, kilib=3, qlib=3, reac=0
[xqt inv
 reset k=khat
[xqt dcu
 copy 1, <slib> u0
 copy 1, <slib> ud0
 copy 3, <slib> stat disp
 change <slib> u0 mask mask mask stat disp 0 1
 change <slib> ud0 mask mask mask ud vec 0 1
 change <slib> stat mask 1 1 udd vec 0 1
 toc 1
 toc <slib>
*close 3 /delete
[xqt vprt
 lib = <slib>
 format = 4
 print stat disp 0 ' initial displacement vector
 print ud vec 0 ' initial velocity vector
 print udd vec 0 ' initial acceleration vector
.
. iterate for "nstep" time steps
.
[xqt aus
*def/i pcnt = 1
*do $step = 0,<nstep>
  inlib = 21 : outlib = 21
```

Fig. 2-3

```
  define u = <slib> stat disp <$step>
  define ud = <slib> ud vec <$step>
  define udd = <slib> udd vec <$step>
  define m = 1 <mname>
.
  r1 = sum(<a0> u    <a2> ud)
  r2 = sum(<a3> udd       r1)
*def/i stp1 =  <<$step> + 1>
  outlib = <slib>
   applied force <stp1> = prod(m, r2)
[xqt ssol
  reset k=khat, set=<stp1>, qlib=<slib>, reac=0
[xqt aus
  inlib = 21 : outlib = 21
  define utdt = <slib> stat disp <stp1>
  define ut   = <slib> stat disp <$step>
  define udt  = <slib> ud   vec  <$step>
  define uddt = <slib> udd  vec  <$step>
  u1 = sum(utdt -1.0 ut)
  u2 = sum(<a0> u1 <ma2> udt)
  u3 = sum(udt <a6> uddt)
   outlib = <slib>
   udd vec <stp1> = sum(u2, <ma3> uddt)
  define utt = <slib> udd vec <stp1>
   ud vec <stp1> = sum(u3 <a7> utt)
*show/macro pcnt
*if <<pcnt> /eq [pfreq]> /then
100 print every pfreq'th solution
.
[xqt vprt
 lib = <slib>
 format = 4
 print stat disp <stp1> ' displacement vector
*def/i pcnt = 1
[xqt aus
*else
*def/i pcnt = <<pcnt> + 1>
*endif
*enddo
*end

*call CANT_BEAM
#call VIBR MODES (nmodes=4)
#call TR_MODAL (nmodes=4)
*call TR DIRECT
[xqt exit
```

Fig. 2-4

Figure 3. CSM Focus Problem 1

```
$ DEL MESH_FOCUS.DAT;*,MATDAT.DAT;*,PLOT_SPEC.DAT;*,MAIN.DAT;*,PANEL*
   .DAT;*
$ DEL *.L16;*
$ DEL FOCUS.LO1;*
$ NICESPAR
*open 1 focus.101
*set echo=off
.
.
.                  CSM FOCUS PROBLEM 1 --- BUCKLING OF A
.              BLADE-STIFFENED PANEL WITH A DISCONTINUOUS STIFFENER
.
.
. Global Macro Definition:
.
*DEF DOSPAR == <TRUE>    .  FALSE for expt. elements, TRUE for SPAR e33, e43
.
.      PROCEDURE MESH_FOCUS:  Set up the TABLEs to be used by CSM1
.
*procedure mesh_focus
[xqt aus
.
. build table of integer user data
.
TABLE(NI=33,NJ=1,itype=0): CSMP FOCS 1 1
 J=1: 4  0  4  16 >  . nnpe, iopt, nrings, nspokes
.
.     Boundary conditions:
.     vuw ruvw
.
      110 000>    .  Edge x=0.0                (Edge 1)
      111 111>    .  Edge y=2*(be + bs)        (Edge 2)
      010 000>    .  Edge x=Al                 (Edge 3)
      111 111>    .  Edge y=0.0                (Edge 4)
      110 000>    .  Corner at (0.,0 .)
      100 000>    .  Corner at (0.,2*(be + bs))
      000 000>    .  Corner at (Al,2*(be + bs))
      010 000>    .  Corner at (Al,0.)
      100 000>    .  Stiffeners at x=0.0
      000 000>    .  Stiffeners at x=Al

. iwall jwall iref jref nelx nele nelbs nels  ifill
.
    1     2     0     0     6     2     2     2      0
.
. build table of floating point user data
.
TABLE(NI=10,NJ=1): CSMP FOCS 1 2
.
.     a    dhole xc    yc    zc   rat    al     be     bs    hs
.
 J=1: 4.0  2.0  2.0   0.0   0.0 0.25   30.0   1.25   4.5   1.4
*END

.
.      PROCEDURE MATDAT -- SET UP TABLES OF MATERIAL PROPERTIES
.
*procedure matdat
[XQT AUS
.
*def g = 3.84615e+6
.
```

```
. E11 NU12 E22 G12 G13 G23 ALPHA1 ALPHA2 WTDEN
.
    TABLE(NI=9,NJ=2): OMB DATA 1 1
.
    I=1,2,3,4,5,6,7,8,9
    J=1: 10.0E+6 .30 10.0E+6 <g> <g> <g> 0.0 0.0 .1
    J=2: 19.e+6 .38 1.89e+6 .93E+6 .93E+6 .93e+6 1.e-4 1.e-4 .01
.
.     FOCUS PROBLEM DATA:
.
. SKIN OF FOCUS PROBLEM
.
    TABLE(NI=3,NJ=25,itype=0):  LAM OMB  1  1
.
    I=1,2,3
    J=1:  1 .0055     45.
    J=2:  1 .0055    -45.
    J=3:  1 .0055      0.
    J=4:  1 .0055      0.
    J=5:  1 .0055    -45.
    J=6:  1 .0055     45.
    J=7:  1 .0055      0.
    J=8:  1 .0055      0.
    J=9:  1 .0055      0.
    J=10: 1 .0055     45.
    J=11: 1 .0055    -45.
    J=12: 1 .0055      0.
    J=13: 1 .0055      0.
    J=14: 1 .0055      0.
    J=15: 1 .0055    -45.
    J=16: 1 .0055     45.
    J=17: 1 .0055      0.
    J=18: 1 .0055      0.
    J=19: 1 .0055      0.
    J=20: 1 .0055     45.
    J=21: 1 .0055    -45.
    J=22: 1 .0055      0.
    J=23: 1 .0055      0.
    J=24: 1 .0055    -45.
    J=25: 1 .0055     45.
.
.
. BLADE STIFFENERS OF FOCUS PROBLEM
.
    TABLE(NI=3,NJ=24,itype=0): LAM OMB  2  1
.
    I=1,2,3
    J=1:  1 .0055     45.0
    J=2:  1 .0055    -45.0
    J=3:  1 .0055      0.0
    J=4:  1 .0055      0.0
    J=5:  1 .0055      0.0
    J=6:  1 .0055      0.0
    J=7:  1 .0055      0.0
    J=8:  1 .0055      0.0
    J=9:  1 .0055      0.0
    J=10: 1 .0055      0.0
    J=11: 1 .0055      0.0
    J=12: 1 .0055      0.0
    J=13: 1 .0055      0.0
    J=14: 1 .0055      0.0
    J=15: 1 .0055      0.0
    J=16: 1 .0055      0.0
    J=17: 1 .0055      0.0
```

Fig. 3-2

```
     J=18: 1 .0055       0.0
     J=19: 1 .0055       0.0
     J=20: 1 .0055       0.0
     J=21: 1 .0055       0.0
     J=22: 1 .0055       0.0
     J=23: 1 .0055     -45.0
     J=24: 1 .0055      45.0
.
     TABLE (NI=3,NJ=1,itype=0):  LAM OMB  3  1
           J=1 : 2 .1 0.00
.
.
*end
.
.                PROCEDURE PLOT SPEC  --  Set up plotting specifications
.
*procedure plot_spec
[XQT PLTA
     SPEC 1
          STITLE' GROUP 1 AROUND HOLE
          VIEW=3
          E43 1
     SPEC 2
          STITLE' GROUP 3 PLATE SKIN
          VIEW=3
          E43 3
     SPEC 3
          STITLE' GROUP 2 STIFFENERS OVER HOLE MODEL
          VIEW=-2
          E43 2
     SPEC 4
          STITLE' GROUP 4 LEFT OUTER STIFFENER
          VIEW=-2
          E43 4
     SPEC 5
          STITLE' GROUP 5 CENTER TOP OF HOLE STIFFENER
          VIEW=-2
          E43 5
     SPEC 6
          STITLE' GROUP 6 BOTTOM TOP OF HOLE STIFFENER
          VIEW=-2
          E43 6
     SPEC 7
          STITLE' GROUP 7 RIGHT OUTER STIFFENER
          VIEW=-2
          E43 7
     SPEC 8
          ROTATE -15 3 -45 2 -60 1
          VIEW=3
          ALL
*end
.
.                                          MAIN PROGRAM
*procedure main
[xqt tab
 start 1000                                          . Dummy START card
 title' BLADE STIFFENED COMPOSITE PANEL WITH HOLE
*CALL mesh_focus
[xqt csm1
[xqt tab
*ADD PANEL.PRC
*CALL panel_start
 JLOC
*CALL panel_jloc
```

Fig. 3-3

```
 MATC
 1 1.0 0.3
 CON 1
*CALL panel_bc
*CALL matdat
[xqt lau
     reset key1=-1
[xqt eld
*CALL panel conn
[xqt rseq
     reset maxcon=30
[xqt aus
 sysvec : appl moti
*CALL panel_ad
[xqt e
[xqt eks
[xqt topo
     reset maxsub=8190
[xqt k
[xqt inv
[xqt ssol
[xqt vprt
 format=4
 print stat disp
[xqt gsf
 reset embed=1
*if <dospar> /then
[xqt psf                          . PSF only for SPAR elements
 reset display=2
*endif
[xqt kg
[xqt eig
 reset init=5, prob=buck, nreq=2
[xqt vprt
 format=4
 vector=1,2
 print buck mode
*CALL plot_spec
*end
.
.
.
*call main
[xqt exit

TABLE OF CONTENTS, LIBRARY  1

BLADE STIFFENED COMPOSITE PANEL WITH HOLE

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+  Library  1     File: FOCUS.L01                                           +
+  Form: GAL82    File size:   731035 words    No. of Datasets:   41        +
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Seq#   Date     Time    Lk Records  Processor Dataset name
   1* 05:07:86 16:15:58   0       1  TAB       JDF1.BTAB.1.8
   2* 05:07:86 16:15:59   0       1  TAB       JREF.BTAB.2.6
   3* 05:07:86 16:15:59   0       1  TAB       ALTR.BTAB.2.4
   4  05:07:86 16:16:00   0       1  TAB       NDAL
   5  05:07:86 16:16:13   0       1  AUS       CSMP.FOCS.1.1
   6  05:07:86 16:16:13   0       1  AUS       CSMP.FOCS.1.2
   7  05:07:86 16:16:58   0       1  TAB       JDF1.BTAB.1.8
   8  05:07:86 16:16:59   0       1  TAB       JREF.BTAB.2.6
   9  05:07:86 16:16:59   0       1  TAB       ALTR.BTAB.2.4
  10  05:07:86 16:17:16   0       1  TAB       JLOC.BTAB.2.5
```

Fig. 3-4

11	05:07:86	16:17:17	0	1	TAB	MATC.BTAB.2.2
12	05:07:86	16:17:21	0	1	TAB	CON..1
13	05:07:86	16:17:21	0	1	TAB	QJJT.BTAB.2.19
14	05:07:86	16:17:34	0	1	AUS	OMB.DATA.1.1
15	05:07:86	16:17:35	0	1	AUS	LAM.OMB.1.1
16	05:07:86	16:17:36	0	1	AUS	LAM.OMB.2.1
17	05:07:86	16:17:37	0	1	AUS	LAM.OMB.3.1
18	05:07:86	16:17:59	0	1	LAU	SA.BTAB.2.13
19	05:07:86	16:17:59	0	1	LAU	PROP.BTAB.2.101
20	05:07:86	16:18:16	0	7	ELD	DEF.E43.11.4
21	05:07:86	16:18:31	0	1	ELD	GD.E43.11.4
22	05:07:86	16:18:31	0	1	ELD	GTIT.E43.11.4
23	05:07:86	16:18:32	0	1	ELD	DIR.E43.11.4
24	05:07:86	16:18:32	0	1	ELD	ELTS.NAME
25	05:07:86	16:18:32	0	1	ELD	ELTS.NNOD
26	05:07:86	16:18:32	0	1	ELD	ELTS.ISCT
27	05:07:86	16:18:33	0	1	ELD	NS
28	05:07:86	16:18:48	0	1	RSEQ	JSEQ.BTAB.2.17
29	05:07:86	16:19:08	0	1	AUS	APPL.MOTI.1.1
30	05:07:86	16:19:23	0	344	E	E43.EFIL.11.4
31	05:07:86	16:19:31	0	1	E	DEM.DIAG
32	05:07:86	16:20:59	0	12	TOPO	KMAP..1808.276
33	05:07:86	16:21:03	0	11	TOPO	AMAP..6115.1732
34	05:07:86	16:21:25	0	32	K	K.SPAR.36
35	05:07:86	16:22:20	0	59	INV	INV.K.1
36	05:07:86	16:26:13	0	1	SSOL	STAT.DISP.1.1
37	05:07:86	16:26:28	0	1	SSOL	STAT.REAC.1.1
38	05:07:86	16:27:12	0	344	GSF	STRS.E43.1.1
39	05:07:86	16:28:21	0	32	KG	KG.SPAR.36
40	05:07:86	16:29:31	0	1	EIG	BUCK.EVAL.1.1
41	05:07:86	16:29:31	0	5	EIG	BUCK.MODE.1.1

Fig. 3-5

Figure 4. Finite Element Model for Skewed Plate Example Problem

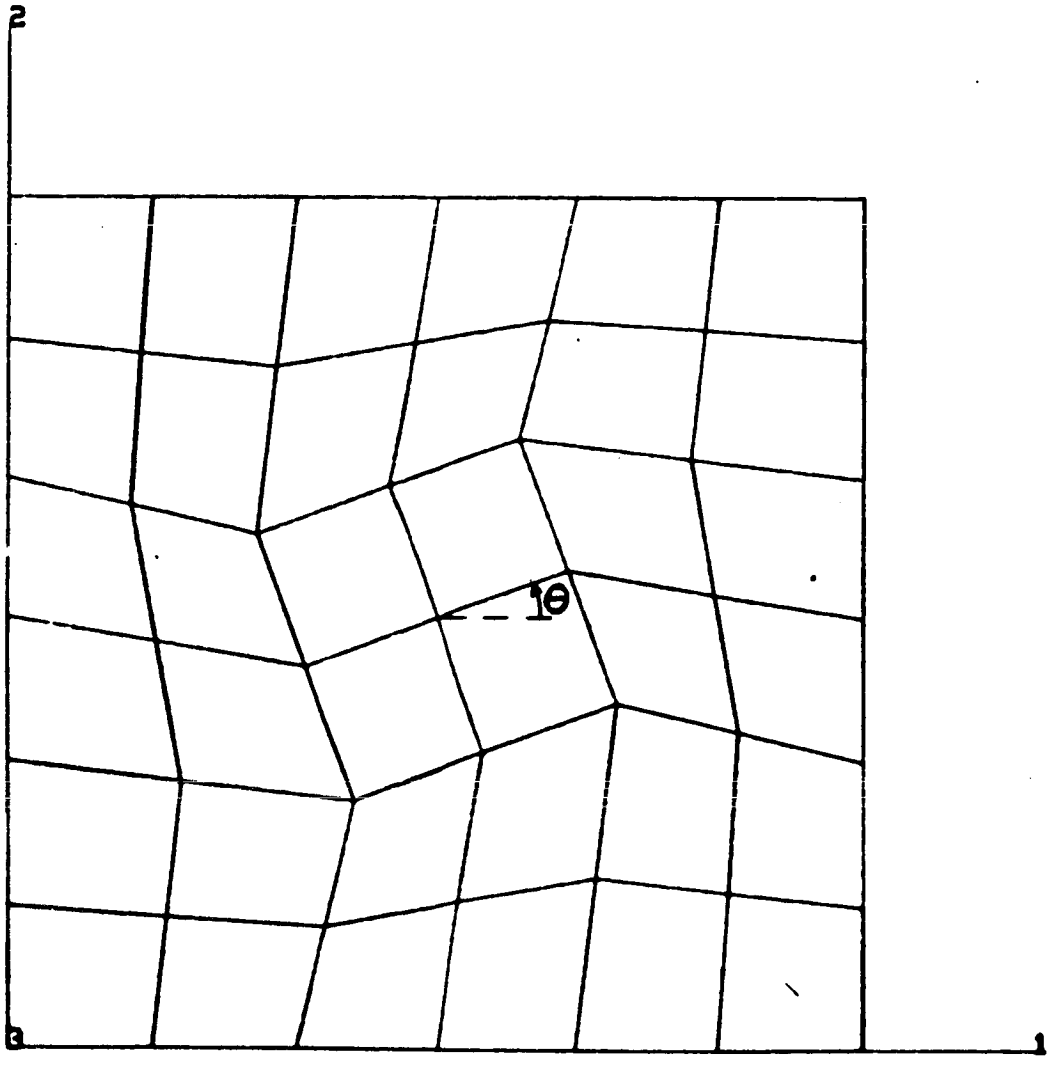


Figure 5. Skewed Plate Example Problem

```
$ delete skew_grid.101;*
$ delete skew_grid.dat;*,matdat.dat;*,g91.dat;*
$ NICESPAR
*set echo=off
.
.                  BUCKLING OF A SIMPLY SUPPORTED SQUARE PLATE
.                 WITH ELEMENTS SKEWED THROUGH AN ANGLE, THETA
.
.   Define Global problem macros:  (These macros are defined here for
.     all procedures.)
.
*DEF IOPT  == 7              . ELEMENT OPTION  (IF DOE43 FALSE)
*DEF NEN   == 9              . NUMBER OF NODES PER ELEMENT
*DEF NEL   == 3              . NUMBER OF (9 NODED) ELEMENTS IN EACH DIRECTION
*DEF NQUAD == 3              . NUMBER OF INTEGRATION POINTS IN EACH DIRECTION
*DEF ANGLE ==20.0            . SKEW ANGLE in degrees
*DEF DL  == 2.5              . LENGTH OF QUARTER MODEL IN EACH DIRECTION
*DEF DOE43 == <FALSE>        . IF FALSE, EXPT. ELEMENTS;  IF TRUE, E43 ELEMENTS
*DEF ISECT == 1              . SECTION PROPERTIES; VALUES MAY BE 1-3 AS FOLLOWS:
.                              1 :  ISOTROPIC (ALUMINUM)
.                              2 :  SKIN OF FOCUS PROBLEM, T=0.00555
.                              3 :  BLADE STIFFENERS OF FOCUS PROB, T=0.00555
.
.      OPEN REQUIRED FILES
.
*open 1, skew_grid.101
.
.      PROCEDURE skew_grid; PERFORMS THE ANALYSIS
.
*procedure skew_grid
.
[XQT TAB
.
.   Some useful definitions:
.
*def/i nn = < <nel>*2 + 1 >                . NUMBER OF NODES IN EACH DIRECTION
*def/i nnod = < <nn>*<nn> >                . TOTAL NUMBER OF NODES
*def/i nnmnx = (<nnod> - <nn> + 1)         . NODE LOCATED AT (0.0,dl)
.
     START <nnod> 6
.
.      If only one nine-noded element is used, define joint locations
.    independent of theta (only used when checking accuracy vs number
.    of nodes per side):
.
*if < <nel> /eq 1 > /then
.
   JOINT LOCATIONS
.
   1          0.0     0.0    0.0  <dl>   0.0    0.0   <nn>   1   <nn>
   <nn>       0.0    <dl>    0.0  <dl>   <dl>   0.0
.
*else
.
.      Set up macros needed for defining joint locations as functions
.      of the skew angle, theta.
.      Some convenient node numbers:
.
*def/i ncen = <(<nnod>+1)%2>               . Center node number
*def/i nt   = < <ncen> + <nn> >            . Node directly above ncen
*def/i nb   = < <ncen> - <nn> >            . Node directly below ncen
*def/i nmidn = < (<nn>+1)%2 >              . Mid side node on y=0 side
```

Fig. 5-1

```
*def/i ntls = < <ncen> + (<nn>)%2 + 1 >    . Node above m.s. node on x=0 side
*def/i nintn = < (<nn>-1)%2 >              . Nr. of nodes between rigid
                                           . center and edge
*def/i nmp1 = < (<nn>+1)%2 + 1 >           . Midside node plus one
.
.    Theta related terms:
.
*def/g theta = < <angle>*<D2R> >           . Skew angle
*def/g st    = <sin(<theta>)>              . Sine of skew angle
*def/g ct    = <cos(<theta>)>              . Cosine of skew angle
*def/g beta = < <pi>/4.-<theta> >          . Convenient angle definition
*def/g sb    = <sin(<beta>)>               .
*def/g cb    = <cos(<beta>)>               .
.
.     Convenient geometric values:
.
*def/g dh    = < <dl>/<nel>/2. >           . Distance between nodes
*def/g h  = < <dl>/2. >                    . Distance to center of 1/4
                                           . model
*def/g a = < 2.^0.5 >                      .
.
.     Convenient sums and products:
.
*def/g dhst = < <dh>*<st> >                .
*def/g dhct = < <dh>*<ct> >                .
*def/g dhsb = < <dh>*<sb> >                .
*def/g dhcb = < <dh>*<cb> >                .
*def/g hpdh = < <h> + <dh> >               .
*def/g hmdh = < <h> - <dh> >               .
*def/g hpct = < <h> + <dh>*<ct> >          .
*def/g hmct = < <h> - <dh>*<ct> >          .
*def/g hpst = < <h> + <dh>*<st> >          .
*def/g hmst = < <h> - <dh>*<st> >          .
*def/g hpcb = < <h> + <dh>*<a>*<cb> >      .
*def/g hmcb = < <h> - <dh>*<a>*<cb> >      .
*def/g hpsb = < <h> + <dh>*<a>*<sb> >      .
*def/g hmsb = < <h> - <dh>*<a>*<sb> >      .
.
   JOINT LOCATIONS
.
.  Center node:
.
   <ncen>          <h>     <h>     0.0
.
.  Nodes between y=0 and rigid center:
.
   <<nmidn>-1>  <hmdh>   0.0    0.0  <hpdh>   0.0    0.0      3      1   <nintn>
   <nn>         <hmsb> <hmcb>  0.0  <hpcb> <hmsb>  0.0
.
.  Nodes between x=0 and rigid center:
.
   <<nb>-<nel>>  0.0   <hmdh>  0.0    0.0   <hpdh>  0.0      3    <nn> <nintn>
   1             <hmsb> <hmcb>  0.0  <hmcb> <hpsb>  0.0
.
.  Nodes between rigid center and y=dly:
.
   <<nt> - 1>   <hmcb> <hpsb>  0.0  <hpsb> <hpcb>  0.0      3      1   <nintn>
   <nn>         <hmdh>  <dl>   0.0  <hpdh>  <dl>   0.0
.
.  Nodes between rigid center and x=dlx:
.
   <<nb> + 1>   <hpcb> <hmsb>  0.0  <hpsb> <hpcb>  0.0      3    <nn> <nintn>
   1             <dl>  <hmdh>  0.0   <dl>  <hpdh>  0.0
.
```

Fig. 5-2

```
.  Nodes on lower left hand corner:
.
   1              0.0     0.0    0.0    0.0   <hmdh>  0.0  <nintn> <nn> <nintn>
   1             <hmdh>   0.0    0.0   <hmsb> <hmcb>  0.0
.
.  Nodes on upper left hand corner:
.
   <ntls>         0.0    <hpdh>  0.0    0.0    <dl>   0.0  <nintn> <nn> <nintn>
   1             <hmcb>  <hpsb>  0.0   <hmdh>  <dl>   0.0
.
.  Nodes on upper right hand corner:
.
   <<nt>+1>      <hpsb>  <hpcb>  0.0   <hpdh>  <dl>   0.0  <nintn> <nn> <nintn>
   1              <dl>   <hpdh>  0.0    <dl>   <dl>   0.0
.
.  Nodes on lower right hand corner:
.
   <nmp1>        <hpdh>   0.0    0.0   <hpcb>  <hmsb>  0.0  <nintn> <nn> <nintn>
   1              <dl>    0.0    0.0    <dl>   <hmdh>  0.0
.
*endif
     MATERIAL CONSTANTS
       1 1.0 .3
.
.      Define constraints.
.
     CONSTRAINT DEFINITION 1                  . Pre-stress; Ny=0., Nx=constant
       symm plane=1                           . Plane 2,3 plane of symmetry
       symm plane=2                           . Plane 1,3 plane of symmetry
       zero 3: 1                              . Constrain center w
       nonzero 1 : <nn>,<nnod>,<nn>           . Apply displacement at x=lx edge
     CONSTRAINT DEFINITION 2                  . Buckling
       symm plane = 1                         . Plane 2,3 plane of symmetry
       symm plane = 2                         . Plane 1,3 plane of symmetry
       zero 1,3,5 : <nnmnx>,<nnod>            . Edge y=ly simply supported
       zero 2,3,4 : <nn>,<nnod>,<nn>          . Edge x=lx simply supported
.
.      Set up material properties using LAU and data supplied by
.      procedure MATDAT
.
*call matdat
[XQT LAU
     reset key1=-1
.
.      Define element connectivity:
.
[XQT ELD
*if < <doe43> /eq <false> > /then
.
.      If experimental elements are to be used, DOE43 is false;  set
.    up the necessary values for these elements
.
*def major = <nen>                            . Number of nodes per element
*def minor = <iopt>                           . Element option
*def nst = <<nquad>*<nquad>*8>                . Number of stress resultants
                                              . per element
     expe EX<MAJOR><MINOR> <MAJOR> <MINOR> <NEN> 6 <NST>   1 101 2
     nsect = 1
     sref = 1
.
*if < <nen> /eq 4 > /then                     . Use 4-noded element mesh generator
*call g41 (nx=<nn>; ny=<nn>; maj=<major>; min=<minor>)
*else
*if < <nen> /eq 9 > /then                     . Use 9-noded element mesh generator
```

Fig. 5-3

```
*call g91 (nx=<nel>; ny=<nel>; maj=<major>; min=<minor>)
*endif
*endif
.
*else
.
.      If DOE43 is true then use E43 SPAR elements.
.
     E43
       nsect = 1
       sref = 1
*def j3 = (<nn>+2)
*def j4 = (<nn>+1)
*def ne43 = (2*<nel>)
       1 2 <j3> <j4> 1 <ne43> <ne43>
*endif
[XQT E
[XQT EKS
[XQT TOPO
[XQT K
.
.      Apply uniform end shortening:
.
[XQT AUS
     sysvec : appl mcti
     i=1:   j=<nn>,<nnod>,<nn>:   -0.10
.
.    Factor:
.
[XQT INV
     reset spdp=2
     online=2
.
.    Solve:
.
[XQT SSOL
.
.    Print static displacements and stresses:
.
[XQT VPRT
     format=4
     print stat disp
[XQT GSF
     reset embed=1
     online=2
*if <doe43> /then
[XQT PSF                    .  PSF only used for standard SPAR elements
     reset display=2
*endif
.
[XQT KG
     online = 1
[XQT INV
     reset con=2
.
.    Compute lowest 2 eigenvalues:
.
[XQT EIG
     reset init=5, nreq=2, prob=stab, con=2
.
.    Print eigenvalues and eigenvectors:
.
[XQT VPRT
     format=4
```

Fig. 5-4

```
     vector=1,2
     print buck mode
*end
.
.      PROCEDURE MATDAT -- SET UP TABLES OF MATERIAL PROPERTIES
.
*procedure matdat
[XQT AUS
.
     TABLE(NI=9,NJ=3): OMB DATA 1 1
.
*def g = 3.84615e+6
. E11 NU12 E22 G12 G13 G23 ALPHA1 ALPHA2 WTDEN
.
     I=1,2,3,4,5,6,7,8,9
     J=1: 20.0E+6 .25 0.050E+6 .30E+6 .25E+6 .25E+6 1.E-4 1.E-4 .01
     J=2: 10.0E+6 .30 10.0E+6 <g> <g> <g> 0.0 0.0 .1
     J=3: 19.e+6 .38 1.89e+6 .93E+6 .93E+6 .93e+6 1.e-4 1.e-4 .01
.
. BUILD LAMINATE DATA TABLES
. MATERIAL TYPE, LAYER THICKNESS, ANGLE IN DEGREES
.
*if < <isect> /eq 1 > /then
.
. ISOTROPIC TEST DATA:
.
     TABLE (NI=3,NJ=1,itype=0):  LAM OMB  1  1
          J=1 : 2 .1 0.00
*endif
*if < <isect> /eq 2 > /then
.
. SKIN OF FOCUS PROBLEM
.
     TABLE(NI=3,NJ=25,itype=0):  LAM OMB  1  1
.
     I=1,2,3
     J=1:  3 .0055     45.
     J=2:  3 .0055    -45.
     J=3:  3 .0055      0.
     J=4:  3 .0055      0.
     J=5:  3 .0055    -45.
     J=6:  3 .0055     45.
     J=7:  3 .0055      0.
     J=8:  3 .0055      0.
     J=9:  3 .0055      0.
     J=10: 3 .0055     45.
     J=11: 3 .0055    -45.
     J=12: 3 .0055      0.
     J=13: 3 .0055      0.
     J=14: 3 .0055      0.
     J=15: 3 .0055    -45.
     J=16: 3 .0055     45.
     J=17: 3 .0055      0.
     J=18: 3 .0055      0.
     J=19: 3 .0055      0.
     J=20: 3 .0055     45.
     J=21: 3 .0055    -45.
     J=22: 3 .0055      0.
     J=23: 3 .0055      0.
     J=24: 3 .0055    -45.
     J=25: 3 .0055     45.
*endif
*if < <isect> /eq 3 > /then
.
```

Fig. 5-5

```
. BLADE STIFFNERS OF FOCUS PROBLEM
.
     TABLE(NI=3,NJ=24,itype=0): LAM OMB  1  1
.
     I=1,2,3
     J=1:  3 .0055     45.0
     J=2:  3 .0055    -45.0
     J=3:  3 .0055      0.0
     J=4:  3 .0055      0.0
     J=5:  3 .0055      0.0
     J=6:  3 .0055      0.0
     J=7:  3 .0055      0.0
     J=8:  3 .0055      0.0
     J=9:  3 .0055      0.0
     J=10: 3 .0055      0.0
     J=11: 3 .0055      0.0
     J=12: 3 .0055      0.0
     J=13: 3 .0055      0.0
     J=14: 3 .0055      0.0
     J=15: 3 .0055      0.0
     J=16: 3 .0055      0.0
     J=17: 3 .0055      0.0
     J=18: 3 .0055      0.0
     J=19: 3 .0055      0.0
     J=20: 3 .0055      0.0
     J=21: 3 .0055      0.0
     J=22: 3 .0055      0.0
     J=23: 3 .0055    -45.0
     J=24: 3 .0055     45.0
*endif
*end
.
.       PROCEDURE G41 -- MESH GENERATOR FOR FOUR-NODED ELEMENTS
.
*procedure g41 (nx; ny; maj; min)
.
*def/i nx = [nx]
*def/i ny = [ny]
*def/i maj = [maj]
*def/i min = [min]
*do $j = 1,<<ny>-1>
    *do $i = 1,<<nx>-1>
        *def n1 = < <nx>*<<$j>-1> + <$i> >
        *def n2 = < <n1> + 1 >
        *def n3 = < <n2> + <nx> >
        *def n4 = < <n1> + <nx> >
         <n1>  <n2>  <n3>  <n4>
    *enddo
*enddo
*end
.
.       PROCEDURE G91 -- MESH GENERATOR FOR NINE-NODED ELEMENTS
.
*procedure g91 (nx; ny; maj; min)
.
*def/i nx = [nx]
*def/i ny = [ny]
*def/i maj = [maj]
*def/i min = [min]
*def/i n1 = 1
*def/i j1 = 1
*def/i iinc = 1
*def jinc = (2*<nx> + 1)
*show macro
```

Fig. 5-6

```
*do $j = 1,<ny>
*do $i = 1,<nx>
*def n5 = (<n1> + <iinc>)
*def n2 = (<n5> + <iinc>)
*def n8 = (<n1> + <jinc>)
*def n9 = (<n5> + <jinc>)
*def n6 = (<n2> + <jinc>)
*def n4 = (<n8> + <jinc>)
*def n7 = (<n9> + <jinc>)
*def n3 = (<n6> + <jinc>)
*remark ex<maj><min> <n1> <n2> <n3> <n4> <n5> <n6> <n7> <n8> <n9>
 <n1> <n2> <n3> <n4> <n5> <n6> <n7> <n8> <n9>
*def n1 = (<n1> + 2*<iinc>)
*end
*def n1 = (<j1> + (2*<$j>*<jinc>))
*enddo
*end
.
.   All procedures have been defined; Call the main procedure and exit.
.
*call skew_grid
[xqt exit

TABLE OF CONTENTS, LIBRARY  1

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+  Library  1      File: SKEW_GRID.LO1                                        +
+  Form: GAL82     File size:   170163 words     No. of Datasets:   34       +
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Seq#    Date      Time     Lk Records  Processor Dataset name
   1  05:07:86 15:54:21   0        1  TAB       JDF1.BTAB.1.8
   2  05:07:86 15:54:21   0        1  TAB       JREF.BTAB.2.6
   3  05:07:86 15:54:21   0        1  TAB       ALTR.BTAB.2.4
   4  05:07:86 15:54:24   0        1  TAB       TEXT.BTAB.2.1
   5  05:07:86 15:54:27   0        1  TAB       JLOC.BTAB.2.5
   6  05:07:86 15:54:27   0        1  TAB       MATC.BTAB.2.2
   7  05:07:86 15:54:28   0        1  TAB       CON..1
   8  05:07:86 15:54:29   0        1  TAB       CON..2
   9  05:07:86 15:54:30   0        1  TAB       QJJT.BTAB.2.19
  10  05:07:86 15:54:35   0        1  AUS       OMB.DATA.1.1
  11  05:07:86 15:54:36   0        1  AUS       LAM.OMB.1.1
  12  05:07:86 15:54:41   0        1  LAU       SA.BTAB.2.13
  13  05:07:86 15:54:41   0        1  LAU       PROP.BTAB.2.101
  14  05:07:86 15:54:49   0        1  ELD       DEF.EX91..9
  15  05:07:86 15:54:55   0        1  ELD       GD.EX91..9
  16  05:07:86 15:54:55   0        1  ELD       GTIT.EX91..9
  17  05:07:86 15:54:56   0        1  ELD       DIR.EX91..9
  18  05:07:86 15:54:56   0        1  ELD       ELTS.NAME
  19  05:07:86 15:54:57   0        1  ELD       ELTS.NNOD
  20  05:07:86 15:54:57   0        1  ELD       ELTS.ISCT
  21  05:07:86 15:54:57   0        1  ELD       NS
  22  05:07:86 15:55:02   0        9  E         EX91.EFIL..9
  23  05:07:86 15:55:13   0        1  E         DEM.DIAG
  24  05:07:86 15:56:11   0        2  TOPO      KMAP..337.85
  25  05:07:86 15:56:12   0        3  TOPO      AMAP..553.153
  26  05:07:86 15:56:16   0        5  K         K.SPAR.25
  27  05:07:86 15:56:33   0        1  AUS       APPL.MOTI.1.1
  28  05:07:86 15:56:40   0        8  INV       INV.K.1
  29  05:07:86 15:57:03   0        1  SSOL      STAT.DISP.1.1
  30  05:07:86 15:57:09   0        1  SSOL      STAT.REAC.1.1
  31  05:07:86 15:57:41   0        5  KG        KG.SPAR.25
  32  05:07:86 15:58:09   0        4  INV       INV.K.2
  33  05:07:86 15:58:36   0        1  EIG       BUCK.EVAL.1.2
  34  05:07:86 15:58:36   0        5  EIG       BUCK.MODE.1.2
```

Fig. 5-7

APPENDIX A. Description of NICE/SPAR Datasets

The contents of this appendix are extracted from Reference 8 with additions and corrections added to reflect testbed usage. Descriptions of many of the data sets that are created and used by NICE/SPAR processors are given, ordered alphabetically by data set name. Each nominal data set name involves four components referred to as NAME1, NAME2, NAME3, and NAME4. NAME1 and NAME2 are alphanumeric names with a maximum of four characters each. NAME3 and NAME4 are integers.

The contents of most of the data sets may be viewed logically as two-dimensional tables, where NI is the first dimension, or column-size, and NJ is the second dimension, or row-size. The data is written to the NICE/SPAR global database as named records in nominal data sets, with a record length of NI*NJ data items. Where the dataset is blocked, each block is written as one nominal record. The NJ parameter (row-size) is stored in the NICE/SPAR record as the matrix dimension parameter. The record name used by the currently installed NICE/SPAR processors is "DATA".

Most data sets contain data of only a single type: integer, single precision real, double precision real, or alphanumeric. These are indicated by type codes 0, -1 or 1, -2 or 2, and 4, respectively. Alphanumeric data is packed four characters to a machine word.

ALTR BTAB 2 4

Created from ALTREF in processor TAB
NJ = Number of alternate reference frames
NI = 12
Type = real

Contents of each entry:

1. a_{11}
2. a_{21}
3. a_{31}
4. a_{12}
5. a_{22}
6. a_{32}
7. a_{13}
8. a_{23}
9. a_{33}

} Components of a coordinate transformation matrix

10. X_o
11. Y_o
12. Z_o

} Location of origin of alternate reference frame given in global coordinates

Formula:

$$\begin{Bmatrix} X_a \\ Y_a \\ Z_a \end{Bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{Bmatrix} X_g \\ Y_g \\ Z_g \end{Bmatrix} + \begin{Bmatrix} X_o \\ Y_o \\ Z_o \end{Bmatrix}$$

coordinates in alternate reference frame — coordinates in global reference frame

AMAP 0 ic2 isize

ic2 = Parameter reflecting the cost of equation solution given a factored system matrix. Computed in processor TOPO.
isize = The maximum number of submatrices involved during the factorization process.

Created in processor TOPO and used by INV to guide factorization of system matrices.
NJ = total number of joints in the model
Type = integer

The purpose of AMAP is to furnish compact information describing the location of each submatrix in the "active" upper triangle of the system matrix as each joint is eliminated. During factorization the active upper triangle is held in the work array S(JDF, JDF, isize). The pointers in AMAP point to JDF by JDF submatrices in this array. The data set consists of one or more records with the default record size of 1792 words. A joint group is included for each joint in the model. Each record contains the following:

JOINTS - The number of joint groups contained in this record. A joint group is not allowed to span a record boundary.

Repeated JOINTS times:

JNT - The number of the current joint.

CONRNG - The number of submatrices including the diagonal in the upper triangle for the current joint as it is being eliminated.

CONECT(CONRNG-1) - A list of column positions for each of the submatrices in the JNT row.

SUBMAP (CONRNG * (CONRNG + 1)/2) - Contains a pointer into the work array, S, for each submatrix in the active upper triangle.

APPL FORC iset 1

iset Load set
Created in processor AUS.
NJ total number of joints in the model
Type real single precision

SYSVEC format data set. A SYSVEC data set has NJ equal to the number of joints in the model and NI equal to the number of active (unconstrained) degrees of freedom per joint. When these data sets are manipulated by a processor, they are expanded to 6 degrees of freedom per joint by subroutine PUP. SYSVEC format data sets frequently have multiple blocks. The meaning of the block number varies depending on the particular data set. In static analysis the block number indicates the load case. In eigenvalue problems the block number indicates the eigenvector.

Contents:

Each entry contains applied forces and moments on that joint in each active direction.

APPL MOTI iset 1

iset = Load set
Created in processor AUS.
SYSVEC format. See "APPL FORC iset".

Contents:

Each entry contains applied motions on that joint in each active direction.

BA BTAB 2 9

Created from E21 section properties in processor TAB
NJ = Number of entries
NI = 31
Type = real

Contents of each entry:

(See reference 5 for description of DSY input of E21 section properties.)

1. Element type indicator
2. Not used
3. Not used
4. I_1
5. α_1
6. I_2
7. α_2
8. a
9. f
10. f_1
11. z_1
12. z_2
13. Θ
14. q_1
15. q_2
16. q_3
17. Number of points at which stresses are to be calculated
18. y_{11}
19. y_{12}
20. y_{21}
21. y_{22}
22. y_{31}
23. y_{32}
24. y_{41}
25. y_{42}
26. b_1
27. t_1
28. b_2
29. t_2
30. b_3
31. t_3

BC BTAB 2 11

Created from E23 section properties in processor TAB
NJ = Number of entries
NI = 6
Type = real

Contents of each entry:

1. Cross-sectional area of axial element
2. Cross-sectional area of axial element
3. }
4. } Not used.
5. }
6. }

BUCK EVAL iset ncon

iset = Load set
ncon = Constraint case
Created in processor EIG
NJ = 1
NI = Number of eigenvalues
Type = real

Contents:

Buckling eigenvalues corresponding to each eigenvector in "BUCK MODE".

BUCK MODE iset ncon

iset = Load set
ncon = Constraint case
Created in processor EIG.
SYSVEC format. See "APPL FORC iset".

Contents:

Each block of data contains an eigenvector corresponding to an eigenvalue stored in "BUCK EVAL".

CASE TITL iset 1

iset = Load set
Created in processor AUS
Number of blocks = Number of load cases in this load set
Type = alphanumeric

Contents:

Each block contains the title for the corresponding load case in text.

CEM SPAR jdf2 0

jdf2 = square of the number of active degrees of freedom per joint
Created in processor M
A SPAR format system matrix. See "K SPAR jdf2 0"

Contents:

The unconstrained system consistent mass matrix considering only the structural and nonstructural distributed mass associated with the elements.

CON 0 ncon 0

ncon = Constraint case
Created from CON in processor TAB
NJ = Number of joints
NI = 1
Type = integer

Contents:

Each entry contains an integer representing the joint reference frame number and constrained components for that joint. This integer is interpreted as a bit pattern with two bits allocated for each joint degree of freedom and the joint reference frame number stored in the leading bits. For each joint degree of freedom the bit pattern 00 indicates that component is free, the pattern 01 indicates that component is constrained to be zero, and the pattern 10 indicates that a non-zero value of this component will be applied using the APPL MOTI data set.

For example:

A joint with components 1, 2, 3 and 5 zeroed out and JREF = 7 (0111_2 would have the integer 28949_{10} stored according to the following binary bit pattern:

JREF number	Joint Motion Components					
	6	5	4	3	2	1
0...0111	00	01	00	01	01	01

Component 1 (constrained)	=	1	× 1	=	1
Component 2 (constrained)	=	1	× 4	=	4
Component 3 (constrained)	=	1	× 16	=	16
Component 4 (unconstrained)	=	0	× 64	=	0
Component 5 (constrained)	=	1	× 256	=	256
Component 6 (unconstrained)	=	0	× 1024	=	0
JREF number = 7	=	7	× 4096	=	+ 28 672
Integer stored for this joint				=	28 949

DEF "element-name" y z

"element-name"	= element name (e.g., E43, S81, EX97)
y	= element type number (E21=1 thru E44=12, S41=16, S61=17, S81=18, experimental=0)
z	= number of joints per element

Created in processor ELD
NJ = [blocksize/NI] where blocksize is determined by the ELD RESET parameter LREC (default=896)
NI varies depending on the number of nodes in the element
Type = integer

Contents:

1. Element number
2. Group number
3. Element number within group
4. Stress reference frame number
5. N3 of corresponding dataset xx BTAB N3 N4
6. N4 where xx = BA, BB, SA, . . .
7. Index of MATC entry for element material constants
8. Index of section property dataset entry for elem. section properties
9. Index of non-structural weight dataset entry (NSW)
10. Index of rigid link offset dataset entry (BRL)
11. Index of beam orientation dataset entry (MREF)
12. Section type code
13. Node #1
14. Node #2
15. Node #3
16. Node #4

.
.
.

DEM DIAG 0 0

Created in processor E
SYSVEC format. See APPL FORC iset

Contents:

System mass matrix in diagonal form.

DIR "element-name" y z

"element-name"	= element name (e.g., E43, S81, EX97)
y	= element type number (E21=1 thru E44=12, experimental=0)
z	= number of joints per element

Created in processor ELD
NJ = 1
NI = 20
Type = integer

Contents:

1. Number of nodes
2. Element type number
3. Number of elements of this type
4. N4 in name of dataset "xx BTAB N3 N4" where xx is BA, BB, SA. . .
5. Length of E-file entry for this element
6. Offset of the end of segment 1 from the beginning of E-file entry
7. Offset of the end of segment 2 from the beginning of E-file entry
8. Offset of the end of segment 3 from the beginning of E-file entry
9. Offset of the end of segment 4 from the beginning of E-file entry
10. Offset of the end of segment 5 from the beginning of E-file entry
11. Offset of the end of segment 6 from the beginning of E-file entry
12. Offset of the end of segment 7 from the beginning of E-file entry
13. Offset of the end of segment 8 from the beginning of E-file entry
14. Offset of the end of segment 9 from the beginning of E-file entry
15. Precision of element stiffness in segment 5 of the E-file entry;
    1 = single precision, 2 = double precision
16. Number of stresses
17. Number of thermal loads
18. Number of degrees of freedom per node
19. MAJOR (=1 for beams, =2 for plates/shells, =3 for solids)
20. MINOR

DISL Exx iset icase

Exx Element name
iset Load set
icase Load case within load set
Created in processor AUS
NJ Number of elements of this type
Type real

For 2-node elements:
NI 6
Contents of each entry:

1. Displacements in direction 1
2. Displacement in direction 2
3. Displacement in direction 3
4. Rotation about axis 1
5. Rotation about axis 2
6. Rotation about axis 3

These displacements and rotations are relative to a reference frame, parallel to the element's reference frame, and embedded in node 2.

For E31 elements:
NI 3
Contents of each entry:

1. Displacement of joint 2 in direction 1
2. Displacement of joint 3 in direction 1
3. Displacement of joint 3 in direction 2

For E32 elements:
NI 6
Contents of each entry:

1. Displacement of joint 2 in direction 3
2. Rotation of joint 2 about axis 1
3. Rotation of joint 2 about axis 2
4. Displacement of joint 3 in direction 3
5. Rotation of joint 3 about axis 1
6. Rotation of joint 3 about axis 2

DISL Exx iset icase (continued)

For E33 elements

NI = 9

Contents of each entry:

1. Displacement of joint 2 in direction 1
2. Displacement of joint 3 in direction 1
3. Displacement of joint 3 in direction 2
4. Displacement of joint 2 in direction 3
5. Rotation of joint 2 about axis 1
6. Rotation of joint 2 about axis 2
7. Displacement of joint 3 in direction 3
8. Rotation of joint 3 about axis 1
9. Rotation of joint 3 about axis 2

For E41 elements:

NI = 6

Contents of each entry:

1. Displacement of joint 2 in direction 1
2. Displacement of joint 3 in direction 1
3. Displacement of joint 3 in direction 2
4. Displacement of joint 4 in direction 1
5. Displacement of joint 4 in direction 2
6. Displacement of joint 4 in direction 3

For E42 elements:

NI = 6

Contents of each entry:

1. Displacement of joint 2 in direction 3
2. Rotation of joint 2 about axis 1
3. Rotation of joint 2 about axis 2
4. Displacement of joint 3 in direction 3
5. Rotation of joint 3 about axis 1
6. Rotation of joint 3 about axis 2
7. Displacement of joint 4 in direction 3
8. Rotation of joint 4 about axis 1
9. Rotation of joint 4 about axis 2

DISL Exx iset icase (concluded)

For E43 elements:
NI = 14
Contents of each entry:

1. Displacement of joint 2 in direction 1
2. Displacement of joint 3 in direction 1
3. Displacement of joint 3 in direction 2
4. Displacement of joint 4 in direction 1
5. Displacement of joint 4 in direction 2
6. Displacement of joint 2 in direction 3
7. Rotation of joint 2 about axis 1
8. Rotation of joint 2 about axis 2
9. Displacement of joint 3 in direction 3
10. Rotation of joint 3 about axis 1
11. Rotation of joint 3 about axis 2
12. Displacement of joint 4 in direction 3.
13. Rotation of joint 4 about axis 1
14. Rotation of joint 4 about axis 2

For E44 elements:
NI = 6
Contents of each entry:

1. Displacement of joint 2 in direction 1
2. Displacement of joint 3 in direction 1
3. Displacement of joint 3 in direction 2
4. Displacement of joint 4 in direction 1
5. Displacement of joint 4 in direction 2
6. Displacement of joint 4 in direction 3

"element-name" EFIL y z

"element-name"	= element name (e.g., E43, S81, EX97)
y	= element type number (E21=1 thru E44=12, experimental=0)
z	= number of joints per element

Created by processor E, modified by processor GSF
NJ = Number of elements of this type
Type = mixed integer and real

Contents:

The EFIL dataset contains NJ entries, written as one entry per nominal record of mixed type data; each entry is made up of segments whose offsets from the beginning of the entry may be determined from the DIR dataset for the corresponding element type:

Segment 1. Integer information, same as DEF dataset entry for this element

Segment 2. Material segment

Segment 3. Geometry segment (length is element dependent)

Segment 4. Property segment

Segment 5. Intrinsic stiffness segment

Segment 6. Stress recovery segment

Segment 7. Stress state segment

Segment 8. Thermal force segment

Segment 9. Thermal stress recovery segment

The geometry segment (seg. 3) is made up of blocks of real data which vary depending on the element type as follows:

SPAR beam elements:

word 1	: Z
words 2-4	: DIJ(3)
words 5-13	: R(3,3)
words 14-31	: Q (3,3,2)
words 32-37	: XOFF(3,2)

SPAR 3-Node Plate Elements:

word 1	: AREA
words 2-7	: X(2,3)
words 8-43	: T(3,3,4)

SPAR 4-Node Plate Elements:

word 1	: AREA
word 2	: A123
word 3	: A124
word 4	: X34 (amount of warping)
words 5-12	: X(2,4)
words 13-57	: T(3,3,5)

SPAR Solid Elements:

word 1	: VOL
word 2 thru (1+3*number of nodes)	: X(3,number of nodes)
words (2+3*number of nodes) thru (10+12*number of nodes)	: T(3,3,number of nodes+1)

ELTS ISCT 0 0

Created in processor ELD.
NJ = Number of element types in the model
NI = 1
Type = integer

Contents:

N4 of "xx BTAB N3 N4" where xx = BA,BC,SA, . . . which contains section property information for an element type.

ELTS NAME 0 0

Created in processor ELD.
NI = 1
NJ = Number of element types in the model
Type = alphanumeric

Contents:

Alphanumeric element name of each element used in the model.

ELTS NNOD 0 0

Created in processor ELD.
NJ = Number of element types in the model
NI = 1
Type = integer

Contents:

The number of nodes in each element type.

GD Exx y z

Exx = Element name
y = Type number (E21 = 1 through E44 = 12, S41=16, S61=17, S81=18)
z = Number of joints/element
Created from element definitions in processor ELD.
NJ = Number of groups
NI = 2
Type = integer

Contents of each entry:

1. Total number of elements within group
2. Cumulative total of elements in all previous groups

GSTR E31 iset icase

Contains stress resultants transformed to the global reference frame.

iset = Load set
icase = Load case within set
Created in processor GSF.
NJ = Number of E31 elements
NI = 11
Type = real

The dataset contains NJ nominal records, NI items per record.

Contents of each record: (Note - x, y, z are in global reference frame.)

1. Group number
2. Element number within group
3. Joint #1
4. Joint #2
5. Joint #3
6. T_{11}
7. T_{22}
8. T_{22}
9. Tractive force in x-direction N_x
10. Tractive force in y-direction N_y
11. Shearing force N_{xy}

Formulae:

$$S_x = N_x/\text{thickness}$$
$$S_y = N_y/\text{thickness}$$
$$T_{xy} = N_{xy}/\text{thickness}$$

GSTR E32 iset icase

Contains stress resultants transformed to the global reference frame.

iset = Load set
icase = Load case within set
Created in processor GSF.
NJ = Number of E32 elements
NI = 28
Type = real

The dataset contains NJ nominal records, NI items per record.

Contents of each record:

1. Group number
2. Element number within group
3. Joint #1
4. Joint #2
5. Joint #3
6. Not used
7. Index of section property dataset entry for element section properties
8. Section type code
9. M_x Bending moment about-axis at joint 1
10. M_y Bending moment about y-axis at joint 1
11. M_{xy} Twisting moment at joint 1
12. Q_x Transverse shear in x-direction at joint 1
13. Q_y Transverse shear in y-direction at joint 1
14. M_x Bending moment about x-axis at joint 2
15. M_y Bending moment about y-axis at joint 2
16. M_{xy} Twisting moment at joint 2
17. Q_x Transverse shear in x-direction at joint 2
18. Q_y Transverse shear in y-direction at joint 2
19. M_x Bending moment about x-axis at joint 3
20. M_y Bending moment about y-axis at joint 3
21. M_{xy} Twisting moment at joint 3
22. Q_x Transverse shear in x-direction at joint 3
23. Q_y Transverse shear in y-direction at joint 3

GSTR E32 iset icase (concluded)

24. M_x Bending moment about x-axis at the center
25. M_y Bending moment about y-axis at the center
26. M_{xy} Twisting moment at the center
27. Q_x Transverse shear in x-direction at the center
28. Q_y Transverse shear in y-direction at the center

Formulae:

$$S_x = f_{4j} M_x \qquad f_{ij} = 1/\text{thickness for } i \text{ and } j = 1, 2, 3$$
$$S_y = f_{5j} M_y \qquad f_{42} = f_{52} = -f_{62} = 6/(\text{thickness})^2$$
$$T_{xy} = f_{6j} M_{xy} \qquad f_{43} = f_{53} = -f_{63} = -6/(\text{thickness})^2$$

GSTR E33 iset icase

Contains stress resultants transformed to the global reference frame.

iset = Load set
icase = Load case within set
Created in processor GSF.
NJ = Number of E33 elements
NI = 31
Type = real

The dataset contains NJ nominal records, NI items per record.

Contents of each record:

1. Group number
2. Element number within group
3. Joint #1
4. Joint #2
5. Joint #3
6. Not used
7. Index of section property dataset entry for element section properties
8. Section type code
9. N_x Tractive force in x-direction
10. N_y Tractive force in y-direction
11. N_{xy} Shearing force
12. M_x Bending moment about x-axis at joint 1
13. M_y Bending moment about y-axis at joint 1
14. M_{xy} Twisting moment at joint 1
15. Q_x Transverse shear in x-direction at joint 1
16. Q_y Transverse shear in y-direction at joint 1
17. M_x Bending moment about x-axis at joint 2
18. M_y Bending moment about y-axis at joint 2
19. M_{xy} Twisting moment at joint 2
20. Q_x Transverse shear in x-direction at joint 2
21. Q_y Transverse shear in y-direction at joint 2
22. M_x Bending moment about x-axis at joint 3
23. M_y Bending moment about y-axis at joint 3

GSTR E33 iset icase (concluded)

24. M_{xy} Twisting moment at joint 3
25. Q_x Transverse shear in x-direction at joint 3
26. Q_y Transverse shear in y-direction at joint 3
27. M_x Bending moment about x-axis at the center
28. M_y Bending moment about y-axis at the center
29. M_{xy} Twisting moment at the center
30. Q_x Transverse shear in x-direction at the center
31. Q_y Transverse shear in y-direction at the center

Formulae:

$$S_x = f_{1j}N_x + f_{4j}M_x \qquad f_{ij} = 1/\text{thickness for } i \text{ and } j = 1,2,3$$
$$S_y = f_{2j}N_y + f_{5j}M_y \qquad f_{42} = f_{52} = -f_{62} = 6/(\text{thickness})^2$$
$$T_{xy} = f_{3j}N_{xy} + f_{6j}M_{xy} \qquad f_{43} = f_{53} = -f_{63} = -6/(\text{thickness})^2$$

GSTR E41 iset icase

Contains stress resultants transformed to the global reference frame.

iset = Load set
icase = Load case within set
Created in processor GSF.
NJ = Number of E41 elements
NI = 23
Type = real

The dataset contains NJ nominal records, NI items per record.

Contents of each record:

1. Group number
2. Element number within group
3. Joint #1
4. Joint #2
5. Joint #3
6. Joint #4
7. Index of section property dataset entry for element section properties
8. Section type code
9. N_x Tractive force in x-direction at joint 1
10. N_y Tractive force in y-direction at joint 1
11. N_{xy} Shearing force at joint 1
12. N_x Tractive force in x-direction at joint 2
13. N_y Tractive force in y-direction at joint 2
14. N_{xy} Shearing force at joint 2
15. N_x Tractive force in x-direction at joint 3
16. N_y Tractive force in y-direction at joint 3
17. N_{xy} Shearing force at joint 3
18. N_x Tractive force in x-direction at joint 4
19. N_y Tractive force in y-direction at joint 4
20. N_{xy} Shearing force at joint 4
21. N_x Tractive force in x-direction at the center
22. N_y Tractive force in y-direction at the center
23. N_{xy} Shearing force at the center

GSTR E41 iset icase (concluded)

Formulae:

$$S_x = N_x/\text{thickness}$$
$$S_y = N_y/\text{thickness}$$
$$T_{xy} = N_{xy}/\text{thickness}$$

GSTR E42 iset icase

Contains stress resultants transformed to the global reference frame.

iset = Load set
icase = Load case within set
Created in processor GSF.
NJ = Number of E42 elements
NI = 33
Type = real

The dataset contains NJ nominal records, NI items per record.

Contents of each record:

1. Group number
2. Element number within group
3. Joint #1
4. Joint #2
5. Joint #3
6. Joint #4
7. Index of section property dataset entry for element section properties
8. Section type code
9. M_x Bending moment about x-axis at joint 1
10. M_y Bending moment about y-axis at joint 1
11. M_{xy} Twisting moment at joint 1
12. Q_x Transverse shear in x-direction at joint 1
13. Q_y Transverse shear in y-direction at joint 1
14. M_x Bending moment about x-axis at joint 2
15. M_y Bending moment about y-axis at joint 2
16. M_{xy} Twisting moment at joint 2
17. Q_x Transverse shear in x-direction at joint 2
18. Q_y Transverse shear in y-direction at joint 2
19. M_x Bending moment about x-axis at joint 3
20. M_y Bending moment about y-axis at joint 3
21. M_{xy} Twisting moment at joint 3
22. Q_x Transverse shear in x-direction at joint 3
23. Q_y Transverse shear in y-direction at joint 3

GSTR E42 iset icase (concluded)

24. M_x Bending moment about x-axis at joint 4
25. M_y Bending moment about y-axis at joint 4
26. M_{xy} Twisting moment at joint 4
27. Q_x Transverse shear in x-direction at joint 4
28. Q_y Transverse shear in y-direction at joint 4
29. M_x Bending moment about x-axis at the center
30. M_y Bending moment about y-axis at the center
31. M_{xy} Twisting moment at the center
32. Q_x Transverse shear in x-direction at the center
33. Q_y Transverse shear in y-direction at the center

Formulae:

$$S_x = f_{4j} M_x \qquad f_{42} = f_{52} = -f_{62} = -6/(\text{thickness})^2$$
$$S_y = f_{5j} M_y \qquad f_{43} = f_{53} = -f_{63} = -6/(\text{thickness})^2$$
$$T_{xy} = f_{6j} M_{xy}$$

GSTR E43 iset icase

Contains stress resultants transformed to the global reference frame.

iset = Load set
icase = Load case within set
Created in processor GSF
NJ = Number of E43 elements
NI = 48
Type = real

The dataset contains NJ nominal records, NI items per record.

Contents of each record:

1. Group number
2. Element number within group
3. Joint #1
4. Joint #2
5. Joint #3
6. Joint #4
7. Index of section property dataset entry for element section properties
8. Section type code
9. N_x Tractive force in x-direction at joint 1
10. N_y Tractive force in y-direction at joint 1
11. N_{xy} Shearing force at joint 1
12. N_x Tractive force in x-direction at joint 2
13. N_y Tractive force in y-direction at joint 2
14. N_{xy} Shearing force at joint 2
15. N_x Tractive force in x-direction at joint 3
16. N_y Tractive force in y-direction at joint 3
17. N_{xy} Shearing force at joint 3
18. N_x Tractive force in x-direction at joint 4
19. N_y Tractive force in y-direction at joint 4
20. N_{xy} Shearing force at joint 4
21. N_x Tractive force in x-direction at the center
22. N_y Tractive force in y-direction at the center
23. N_{xy} Shearing force at the center

GSTR E43 iset icase (concluded)

24. M_x Bending moment about x-axis at joint 1
25. M_y Bending moment about y-axis at joint 1
26. M_{xy} Twisting moment at joint 1
27. Q_x Transverse shear in x-direction at joint 1
28. Q_y Transverse shear in y-direction at joint 1
29. M_x Bending moment about x-axis at joint 2
30. M_y Bending moment about y-axis at joint 2
31. M_{xy} Twisting moment at joint 2
32. Q_x Transverse shear in x-direction at joint 2
33. Q_y Transverse shear in y-direction at joint 2
34. M_x Bending moment about x-axis at joint 3
35. M_x Bending moment about y-axis at joint 3
36. M_{xy} Twisting moment at joint 3
37. Q_x Transverse shear in x-direction at joint 3
38. Q_y Transverse shear in y-direction at joint 3
39. M_x Bending moment about x-axis at joint 4
40. M_y Bending moment about y-axis at joint 4
41. M_{xy} Twisting moment at joint 4
42. Q_x Transverse shear in x-direction at joint 4
43. Q_y Transverse shear in y-direction at joint 4
44. M_x Bending moment about x-axis at the center
45. M_y Bending moment about y-axis at the center
46. M_{xy} Twisting moment at the center
47. Q_x Transverse shear in x-direction at the center
48. Q_y Transverse shear in y-direction at the center

Formulae:

$$S_x = f_{1j}N_x + f_{4j}M_x \qquad f_{ij} = 1/\text{thickness for } i \text{ and } j = 1,2,3$$
$$S_y = f_{2j}N_y + f_{5j}M_y \qquad f_{42} = f_{52} = -f_{62} = 6/(\text{thickness})^2$$
$$T_{xy} = f_{3j}N_{xy} + f_{6j}M_{xy} \qquad f_{43} = f_{53} = -f_{63} = 6/(\text{thickness})^2$$

GTIT Exx y z

Exx = Element name
y = Type number
(E21 = 1 through E44 = 12, S41=16, S61=17, S81=18 experimental = 0)
z = Number of joints/element
Created from element definitions in processor ELD.
NJ = Number of groups
NI = 15
Type = alphanumeric

Contents of each entry:
15 words of title for each group.
Default is blanks.

INV name ncon 0

name = name of the unfactored SPAR type matrix
ncon = constraint case applied during factorization
Created in processor INV
NJ = total number of joints in the model
Type = mixed real and integer

Contains the upper triangle of the factored system matrix. For a matrix, **A**, factored into the product LDL^T, INV A 2 0 dataset contains the inverse of the diagonal matrix D and the triangular matrix L^T stored by row for the system matrix **A** subject to constraint set 2. The data set consists of one or more records with the default record size of 3584 words. A joint group is included for every joint in the model. Each record contains the following:

JOINTS - The number of joint groups contained in this record. A joint group is not allowed to span a record boundary.

INDEX(JMAX) - An array of integers pointing to the beginning of each joint group in the record. JMAX is defaulted to 50 and JOINTS must be $\leq$ JMAX.

Repeated JOINTS times:

JNT - The number of the current joint.

NZERO - the number of active degrees of freedom at the current joint. If NZERO equals zero, the next joint group follows.

MAP(NZERO) - a list of the unconstrained degrees of freedom at the current joint.

CONRNG - the number of joints connected to the current joint at the time of its elimination.

CONECT(CONRNG-1) - a list of all connected joints in the upper triangle of the factored matrix.

A(JDF, CONRNG, NZERO) - contains the $1/D_{ii}$ and L^T_{ij} components of the factored matrix for this joint. For each active degree of freedom (1 to NZERO) there is a vector JDF $\times$ CONRNG in length which represents a row of the factored matrix.

JDF1 BTAB 1 8

Created from TAB processor START card.
NJ = 1
NI = 18
Type = integer

Contents:

1. Total number of joints.
2. Number of active (unconstrained) degrees of freedom per joint.
3. Number of joint translational degrees of freedom not constrained.
4. A list of unconstrained joint degrees of freedom, filled in consecutively from position number 4; unused values are zero.
5. Example of d.o.f. 1, 2, and 6 unconstrained:
6. 1,2,6,0,0,0
7. A list specifying the order of each unconstrained degree of freedom; zero if not active.
8. Example for d.o.f. 1, 2, and 6 unconstrained:
9. 1,2,0,0,0,3
10. Not used.
11. Not used.
12. Not used.

JLOC BTAB 2 5

Created from JLOC in processor TAB.
NJ = Number of joints
NI = 3
Type = real

Contents:

J = 1, 2, . . . Number of joints

I = 1	X_1 X_2 ... Rectangular coordinates
2	Y_1 Y_2 ... of each joint in the
3	Z_1 Z_2 ... global reference frame

JREF BTAB 2 6

Created by subprocssor JREF in processor TAB.
NJ = Total number of joints
NI = 1
Type = integer

Contents:

Contains the joint reference frame number for each joint, corresponding to the entry in dataset ALTR BTAB 2 4 which contains the definition of each joint reference frame.

JSEQ BTAB 2 17

Created by subprocessor JSEQ in processor TAB or by automatic joint ordering processors.

NJ = number of joints in the model
NI = 1
Type = integer

Contents:

The j^{th} entry contains the elimination order number for joint j.

K SPAR jdf2 0

jdf2 = square of the number of degrees of freedom in the model, JDF.
Created in processor K.
NJ = total number of joints in the model
Type = single or double precision real

Contents:

Contains the assembled global stiffness matrix in the SPAR sparse matrix format. The SPAR sparse matrix format stores only the nonzero JDF by JDF submatrices in the upper triangle of the symmetric system matrix. Submatrix i, j is nonzero if an element connects joints i and j. A SPAR sparse matrix format data set consists of one or more fixed length records with a default record size of 2240 words. A joint group is included for every joint in the model starting in record 1 with the first joint to be eliminated in factorization. Integer information is converted to the numeric type of the dataset before being stored in the record. Each record contains the following:

JOINTS - The number of joint groups contained in this record. A joint group is not allowed to span a record boundary.

Repeated JOINTS times:

CONRNG - The number of submatrices including the diagonal in the upper triangle for the current joint.

SUBMAP(CONRNG) - A list of joints connected to the current joint (listed first).

S(JDF, JDF, CONRNG) - The submatrices in the upper triangle of the system matrix connected to the current joint. These correspond to the joints listed in SUBMAP.

KMAP 0 nsubs ksize

nsubs = the total number of submatrices in a SPAR system matrix for this model

ksize = the maximum number of joints active at any time during the assembly of the system matrix

Created in processor TOPO and used by various processors to guide the assembly of system matrices.

NJ = total number of joints in the model
Type = integer

Contents:

The purpose of KMAP is to furnish compact information about elements connected to each joint in the model. It also defines which upper triangle submatrices will be nonzero for each joint. During assembly of a SPAR system matrix, a work area S(JDF, JDF, KSIZE) is used to hold the active submatrices. Information in KMAP shows where each piece of an elemental matrix fits in the array S. Other information in KMAP shows which joint pair i, j is associated with each submatrix in S.

KMAP 0 nsubs ksize (concluded)

KMAP consists of one or more fixed length records with a default record size of 1792. A joint group is included for each joint in the model. Each joint group contains element groups for elements connected to the joint. Each record contains the following:

JOINTS - Number of joint groups contained in this record. A joint group is not allowed to span a record boundary.

Repeated JOINTS times:

JNT - The number of the current joint

LRNG - Number of elements which connect to JNT and any higher numbered joints. These are elements which will contribute to the upper triangle of the system matrix.

Repeated LRNG times:

NODES - Number of nodes in the current element type

LTYPE - Integer number for this element type

NSE - Element number. For each element type, this number begins at one and increments for each element of the particular type.

ITYPE - Pointer into data set "NS 0 0" for this element type

NSCT - N4 of the section property data set name for this element type. See data set "ELTS ISCT 0 0".

ISCT - Index of section property data set entry for element section properties.

MAP ($NODES * (NODES + 1)/2$) - Location in the work area, S, where each submatrix in the elemental submatrix is to be summed. If MAP(I) $\leq$ 0, then the transpose of the elemental submatrix should be summed into S.

CONRNG - The number of submatrices including the diagonal in the upper triangle for the current joint.

CONECT(CONRNG-1) - A list of joints connected to the current joint.

SUBMAP(CONRNG) - A pointer into the work array, S, for each submatrix associated with the current joint.

LAM OMB nsect 1

Created using AUS/TABLE

Contains the laminate data for shell section number "nsect".

NI = 3
NJ = Number of layers
Type = Real

Contents of each entry:

1. Material number
2. Layer thickness
3. Layer orientation

Repeated NJ times.

LAM O3D nsect 1

Created using AUS/TABLE

Contains the laminate data for solid section number "nsect".

NI = 3
NJ = Number of layers
Type = Real

Contents of each entry:

1. Material number
2. Layer thickness
3. Layer orientation

Repeated NJ times.

MATC BTAB 2 2

Created from MATC in processor TAB.
NJ = Number of material types
NI = 10
Type = real

Contents of each entry:

1. E = Modulus of elasticity
2. ν = Poisson's Ratio
3. $G = E/(2(1+\nu))$
4. ρ = Weight per unit volume
5. α_1 = Thermal expansion coefficient, direction x
6. α_2 = Thermal expansion coefficient, direction y
7. Θ = Angle between element reference frame and the frame used for input of α_1 and α_2.
8. Not used.
9. Not used.
10. Not used.

MREF BTAB 2 7

Created from MREF in processor TAB.
NJ = Number of beam orientation entries
NI = 5
Type = real

Contents of each entry:

Format 1 (Default)

1. Beam axis NB
2. Global axis NG
3. 1. if cosine between NB and NG is positive, -1. if negative
4. Cosine of angle between NB and NG
5. 1. (indicating format = 1)

Format 2

1. X1
2. X2
3. X3
4. I1 axis orientation
5. -1. (indicating format = 2)

MSTR E31 iset icase

Contains stress resultants transformed to the material reference frame.

iset = Load set
icase = Load case within set
Created in processor GSF.
NJ = Number of E31 elements
NI = 11
Type = real

The dataset contains NJ nominal records, NI items per record.

Contents of each record: (Note - x, y, z are in material reference frame.)

1. Group number
2. Element number within group
3. Joint #1
4. Joint #2
5. Joint #3
6. T_{11}
7. T_{22}
8. T_{22}
9. Tractive force in x-direction N_x
10. Tractive force in y-direction N_y
11. Shearing force N_{xy}

Formulae:

$$S_x = N_x/\text{thickness}$$
$$S_y = N_y/\text{thickness}$$
$$T_{xy} = N_{xy}/\text{thickness}$$

MSTR E32 iset icase

Contains stress resultants transformed to the material reference frame.

iset = Load set
icase = Load case within set
Created in processor GSF.
NJ = Number of E32 elements
NI = 28
Type = real

The dataset contains NJ nominal records, NI items per record.

Contents of each record:

1. Group number
2. Element number within group
3. Joint #1
4. Joint #2
5. Joint #3
6. Not used
7. Index of section property dataset entry for element section properties
8. Section type code
9. M_x Bending moment about-axis at joint 1
10. M_y Bending moment about y-axis at joint 1
11. M_{xy} Twisting moment at joint 1
12. Q_x Transverse shear in x-direction at joint 1
13. Q_y Transverse shear in y-direction at joint 1
14. M_x Bending moment about x-axis at joint 2
15. M_y Bending moment about y-axis at joint 2
16. M_{xy} Twisting moment at joint 2
17. Q_x Transverse shear in x-direction at joint 2
18. Q_y Transverse shear in y-direction at joint 2
19. M_x Bending moment about x-axis at joint 3
20. M_y Bending moment about y-axis at joint 3
21. M_{xy} Twisting moment at joint 3
22. Q_x Transverse shear in x-direction at joint 3
23. Q_y Transverse shear in y-direction at joint 3

MSTR E32 iset icase (concluded)

24. M_x Bending moment about x-axis at the center
25. M_y Bending moment about y-axis at the center
26. M_{xy} Twisting moment at the center
27. Q_x Transverse shear in x-direction at the center
28. Q_y Transverse shear in y-direction at the center

Formulae:

$$S_x = f_{4j} M_x \qquad f_{ij} = 1/\text{thickness for } i \text{ and } j = 1,2,3$$
$$S_y = f_{5j} M_y \qquad f_{42} = f_{52} = -f_{62} = 6/(\text{thickness})^2$$
$$T_{xy} = f_{6j} M_{xy} \qquad f_{43} = f_{53} = -f_{63} = -6/(\text{thickness})^2$$

MSTR E33 iset icase

Contains stress resultants transformed to the material reference frame.

iset = Load set
icase = Load case within set
Created in processor GSF.
NJ = Number of E33 elements
NI = 31
Type = real

The dataset contains NJ nominal records, NI items per record.

Contents of each record:

1. Group number
2. Element number within group
3. Joint #1
4. Joint #2
5. Joint #3
6. Not used
7. Index of section property dataset entry for element section properties
8. Section type code
9. N_x Tractive force in x-direction
10. N_y Tractive force in y-direction
11. N_{xy} Shearing force
12. M_x Bending moment about x-axis at joint 1
13. M_y Bending moment about y-axis at joint 1
14. M_{xy} Twisting moment at joint 1
15. Q_x Transverse shear in x-direction at joint 1
16. Q_y Transverse shear in y-direction at joint 1
17. M_x Bending moment about x-axis at joint 2
18. M_y Bending moment about y-axis at joint 2
19. M_{xy} Twisting moment at joint 2
20. Q_x Transverse shear in x-direction at joint 2
21. Q_y Transverse shear in y-direction at joint 2
22. M_x Bending moment about x-axis at joint 3
23. M_y Bending moment about y-axis at joint 3

MSTR E33 iset icase (concluded)

24. M_{xy} Twisting moment at joint 3
25. Q_x Transverse shear in x-direction at joint 3
26. Q_y Transverse shear in y-direction at joint 3
27. M_x Bending moment about x-axis at the center
28. M_y Bending moment about y-axis at the center
29. M_{xy} Twisting moment at the center
30. Q_x Transverse shear in x-direction at the center
31. Q_y Transverse shear in y-direction at the center

Formulae:

$$S_x = f_{1j}N_x + f_{4j}M_x \qquad f_{ij} = 1/\text{thickness for } i \text{ and } j = 1,2,3$$
$$S_y = f_{2j}N_y + f_{5j}M_y \qquad f_{42} = f_{52} = -f_{62} = 6/(\text{thickness})^2$$
$$T_{xy} = f_{3j}N_{xy} + f_{6j}M_{xy} \qquad f_{43} = f_{53} = -f_{63} = -6/(\text{thickness})^2$$

MSTR E41 iset icase

Contains stress resultants transformed to the material reference frame.

iset = Load set
icase = Load case within set
Created in processor GSF.
NJ = Number of E41 elements
NI = 23
Type = real

The dataset contains NJ nominal records, NI items per record.

Contents of each record:

1. Group number
2. Element number within group
3. Joint #1
4. Joint #2
5. Joint #3
6. Joint #4
7. Index of section property dataset entry for element section properties
8. Section type code
9. N_x Tractive force in x-direction at joint 1
10. N_y Tractive force in y-direction at joint 1
11. N_{xy} Shearing force at joint 1
12. N_x Tractive force in x-direction at joint 2
13. N_y Tractive force in y-direction at joint 2
14. N_{xy} Shearing force at joint 2
15. N_x Tractive force in x-direction at joint 3
16. N_y Tractive force in y-direction at joint 3
17. N_{xy} Shearing force at joint 3
18. N_x Tractive force in x-direction at joint 4
19. N_y Tractive force in y-direction at joint 4
20. N_{xy} Shearing force at joint 4
21. N_x Tractive force in x-direction at the center
22. N_y Tractive force in y-direction at the center
23. N_{xy} Shearing force at the center

MSTR E41 iset icase (concluded)

Formulae:

$$S_x = N_x/\text{thickness}$$
$$S_y = N_y/\text{thickness}$$
$$T_{xy} = N_{xy}/\text{thickness}$$

MSTR E42 iset icase

Contains stress resultants transformed to the material reference frame.

iset = Load set
icase = Load case within set
Created in processor GSF.
NJ = Number of E42 elements
NI = 33
Type = real

The dataset contains NJ nominal records, NI items per record.

Contents of each record:

1. Group number
2. Element number within group
3. Joint #1
4. Joint #2
5. Joint #3
6. Joint #4
7. Index of section property dataset entry for element section properties
8. Section type code
9. M_x Bending moment about x-axis at joint 1
10. M_y Bending moment about y-axis at joint 1
11. M_{xy} Twisting moment at joint 1
12. Q_x Transverse shear in x-direction at joint 1
13. Q_y Transverse shear in y-direction at joint 1
14. M_x Bending moment about x-axis at joint 2
15. M_y Bending moment about y-axis at joint 2
16. M_{xy} Twisting moment at joint 2
17. Q_x Transverse shear in x-direction at joint 2
18. Q_y Transverse shear in y-direction at joint 2
19. M_x Bending moment about x-axis at joint 3
20. M_y Bending moment about y-axis at joint 3
21. M_{xy} Twisting moment at joint 3
22. Q_x Transverse shear in x-direction at joint 3
23. Q_y Transverse shear in y-direction at joint 3

MSTR E42 iset icase (concluded)

24. M_x Bending moment about x-axis at joint 4
25. M_y Bending moment about y-axis at joint 4
26. M_{xy} Twisting moment at joint 4
27. Q_x Transverse shear in x-direction at joint 4
28. Q_y Transverse shear in y-direction at joint 4
29. M_x Bending moment about x-axis at the center
30. M_y Bending moment about y-axis at the center
31. M_{xy} Twisting moment at the center
32. Q_x Transverse shear in x-direction at the center
33. Q_y Transverse shear in y-direction at the center

Formulae:

$$S_x = f_{4j} M_x \qquad f_{42} = f_{52} = -f_{62} = -6/(\text{thickness})^2$$
$$S_y = f_{5j} M_y \qquad f_{43} = f_{53} = -f_{63} = -6/(\text{thickness})^2$$
$$T_{xy} = f_{6j} M_{xy}$$

MSTR E43 iset icase

Contains stress resultants transformed to the material reference frame.

iset = Load set
icase = Load case within set
Created in processor GSF
NJ = Number of E43 elements
NI = 48
Type = real

The dataset contains NJ nominal records, NI items per record.

Contents of each record:

1. Group number
2. Element number within group
3. Joint #1
4. Joint #2
5. Joint #3
6. Joint #4
7. Index of section property dataset entry for element section properties
8. Section type code
9. N_x Tractive force in x-direction at joint 1
10. N_y Tractive force in y-direction at joint 1
11. N_{xy} Shearing force at joint 1
12. N_x Tractive force in x-direction at joint 2
13. N_y Tractive force in y-direction at joint 2
14. N_{xy} Shearing force at joint 2
15. N_x Tractive force in x-direction at joint 3
16. N_y Tractive force in y-direction at joint 3
17. N_{xy} Shearing force at joint 3
18. N_x Tractive force in x-direction at joint 4
19. N_y Tractive force in y-direction at joint 4
20. N_{xy} Shearing force at joint 4
21. N_x Tractive force in x-direction at the center
22. N_y Tractive force in y-direction at the center
23. N_{xy} Shearing force at the center

MSTR E43 iset icase (concluded)

24. M_x Bending moment about x-axis at joint 1
25. M_y Bending moment about y-axis at joint 1
26. M_{xy} Twisting moment at joint 1
27. Q_x Transverse shear in x-direction at joint 1
28. Q_y Transverse shear in y-direction at joint 1
29. M_x Bending moment about x-axis at joint 2
30. M_y Bending moment about y-axis at joint 2
31. M_{xy} Twisting moment at joint 2
32. Q_x Transverse shear in x-direction at joint 2
33. Q_y Transverse shear in y-direction at joint 2
34. M_x Bending moment about x-axis at joint 3
35. M_x Bending moment about y-axis at joint 3
36. M_{xy} Twisting moment at joint 3
37. Q_x Transverse shear in x-direction at joint 3
38. Q_y Transverse shear in y-direction at joint 3
39. M_x Bending moment about x-axis at joint 4
40. M_y Bending moment about y-axis at joint 4
41. M_{xy} Twisting moment at joint 4
42. Q_x Transverse shear in x-direction at joint 4
43. Q_y Transverse shear in y-direction at joint 4
44. M_x Bending moment about x-axis at the center
45. M_y Bending moment about y-axis at the center
46. M_{xy} Twisting moment at the center
47. Q_x Transverse shear in x-direction at the center
48. Q_y Transverse shear in y-direction at the center

Formulae:

$$S_x = f_{1j}N_x + f_{4j}M_x \qquad f_{ij} = 1/\text{thickness for } i \text{ and } j = 1,2,3$$
$$S_y = f_{2j}N_y + f_{5j}M_y \qquad f_{42} = f_{52} = -f_{62} = 6/(\text{thickness})^2$$
$$T_{xy} = f_{3j}N_{xy} + f_{6j}M_{xy} \qquad f_{43} = f_{53} = -f_{63} = 6/(\text{thickness})^2$$

NDAL 0 0 0

Created from TITLE card in processor TAB.
NJ = 1
Type = alphanumeric

Contents:

Library title.

NODA PRES iset 1

iset = Load set
Created using subprocessor TABLE in processor AUS.
NJ = Number of joints
NI = 1
Number of blocks = Number of load cases in this load set.
Type = real

Contents.

Each block of data contains nodal pressures for every joint in the structure. One block corresponds to one load case.

NODA TEMP iset 1

iset = Load set
Created using subprocessor TABLE in processor AUS.
NJ = Number of joints
NI = 1
Number of blocks = Number of load cases in this load set.
Type = real

Contents:

Each block of data contains nodal temperatures for every joint in the structure. One block corresponds to one load case.

NS 0 0 0

Created in processor ELD.
NJ = Number of element types present
NI = 1
Type = integer

Contents of each entry:

1. Offset of the end of segment 1 from the beginning of E-file entry
2. Offset of the end of segment 2 from the beginning of E-file entry
3. Offset of the end of segment 3 from the beginning of E-file entry
4. Offset of the end of segment 4 from the beginning of E-file entry
5. Offset of the end of segment 5 from the beginning of E-file entry
6. Offset of the end of segment 6 from the beginning of E-file entry
7. Offset of the end of segment 7 from the beginning of E-file entry
8. Offset of the end of segment 8 from the beginning of E-file entry
9. Offset of the end of segment 9 from the beginning of E-file entry
10. Precision of element stiffness in segment 5 of the E-file entry
    1 = single precision, 2 = double precision
11. Number of stresses
12. Number of thermal loads
13. Number of degrees of freedom per node
14. MAJOR
15. MINOR

(The contents of each entry are the same as words 6-20 of the DIR dataset for the element type.)

OMB DATA 1 1

Created from AUS/TABLE

Contains the material properties for the 2-D section types.

NI = 9
NJ = Number of materials
Type = Real

Contents of each entry:

1. Young's Modulus, E_1
2. Poisson Ratio, ν_{12} $(E_1\nu_{21} = E_2\nu_{12})$
3. Young's Modulus, E_2
4. Shear Modulus, G_{12}
5. Shear Modulus, G_{13}
6. Shear Modulus, G_{23}
7. Linear Thermal Expansion Coefficients, α_1
8. Linear Thermal Expansion Coefficients, α_2
9. Weight Density (weight per unit area)

Repeated NJ times.

C - 2

O3D DATA 1 1

Created using AUS/TABLE

Contains the material properties for the 3-D section types.

NI = 13
NJ = Number of layers
Type = Real

Contents of each layer:

1. Young's Modulus, E_1
2. Young's Modulus, E_2
3. Young's Modulus, E_3
4. Shear Modulus, G_{12}
5. Shear Modulus, G_{23}
6. Shear Modulus, G_{13}
7. Poisson Ratio, ν_{12} $(E_1\nu_{21} = E_2\nu_{12})$
8. Poisson Ratio, ν_{23} $(E_2\nu_{32} = E_3\nu_{23})$
9. Poisson Ratio, ν_{13} $(E_1\nu_{31} = E_3\nu_{13})$
10. Weight density (weight per unit volume)
11. Linear Thermal Expansion Coefficient, α_1
12. Linear Thermal Expansion Coefficient, α_2
13. Linear Thermal Expansion Coefficient, α_3

Repeated NJ times.

PRES Exx iset icase

Exx = Element name
iset = Load set
icase = Load case within Load set
Created in processor AUS
NJ = Number of elements of this type
Type = real

Not defined for 2-node elements.

For 3-node elements:
NI = 3
Contents of each entry:

1. Pressure at joint 1
2. Pressure at joint 2
3. Pressure at joint 3

For 4-node elements:
NI = 4
Contents of each entry:

1. Pressure at joint 1
2. Pressure at joint 2
3. Pressure at joint 3
4. Pressure at joint 4

PROP BTAB 2 101

Created from shell section properties in processor LAU
NI = 40
NJ = Number of shell sections
Type = real

Contents include the stiffness coefficients for a first-order transverse shear deformation theory.

Contents of each entry:

1. A_{11}
2. A_{21}
3. A_{16}
4. B_{11}
5. B_{12}
6. B_{16}
7. A_{12}
8. A_{22}
9. A_{26}
10. B_{12}
11. B_{22}
12. B_{26}
13. A_{16}
14. A_{26}
15. A_{66}
16. B_{16}
17. B_{26}
18. B_{66}
19. B_{11}
20. B_{12}
21. B_{16}
22. D_{11}
23. D_{12}
24. D_{16}
25. B_{12}
26. B_{22}
27. B_{26}
28. D_{12}
29. D_{22}
30. D_{26}
31. B_{16}
32. B_{26}
33. B_{66}
34. D_{16}
35. D_{26}
36. D_{66}
37. CS_{44}
38. CS_{45}
39. CS_{45}
40. CS_{55}

where A_{ij} are the extensional stiffness coefficients, B_{ij} are the bending-extensional coupling stiffness coefficients, D_{ij} are the bending stiffness coefficients, and CS_{ij} are the transverse shear stiffness coefficients.

PROP BTAB 2 101 (concluded)

These stiffness coefficients relate the force and moment resultants to the middle surface strains and curvatures. That is,

$$\begin{Bmatrix} N_x \\ N_y \\ N_{xy} \\ \hline M_x \\ M_y \\ M_{xy} \end{Bmatrix} = \left[\begin{array}{ccc|ccc} A_{11} & A_{12} & A_{16} & B_{11} & B_{12} & B_{16} \\ A_{12} & A_{22} & A_{26} & B_{12} & B_{22} & B_{26} \\ A_{16} & A_{26} & A_{66} & B_{16} & B_{26} & B_{66} \\ \hline B_{11} & B_{12} & B_{16} & D_{11} & D_{12} & D_{16} \\ B_{12} & B_{22} & B_{26} & D_{12} & D_{22} & D_{26} \\ B_{16} & B_{26} & B_{66} & D_{16} & D_{26} & D_{66} \end{array}\right] \begin{Bmatrix} \epsilon_x^o \\ \epsilon_y^o \\ \gamma_{xy}^o \\ \hline \kappa_x \\ \kappa_y \\ -\kappa_{xy} \end{Bmatrix}$$

and

$$\begin{Bmatrix} Q_y \\ Q_x \end{Bmatrix} = \begin{bmatrix} CS_{44} & CS_{45} \\ CS_{45} & CS_{55} \end{bmatrix} \begin{Bmatrix} \gamma_{xy}^o \\ \gamma_{xz}^o \end{Bmatrix}$$

PROP BTAB 2 21

Created from solid section properties in processor LAU
NI = 31
NJ = Number of solid sections
Type = Real

Contents of each entry:

1. weight density (weight/unit volume)
2. a_{11}
3. a_{21}
4. a_{22}
5. a_{31}
6. a_{32}
7. a_{33}
8. a_{41}
9. a_{42}
10. a_{43}
11. a_{44}
12. a_{51}
13. a_{52}
14. a_{53}
15. a_{54}
16. a_{55}
17. a_{61}
18. a_{62}
19. a_{63}
20. a_{64}
21. a_{65}
22. a_{66}
23. linear thermal expansion coefficient, α_x
24. linear thermal expansion coefficient, α_y
25. linear thermal expansion coefficient, α_z
26. Y_{xx}, reference stress for use in stress display
27. Y_{yy}, reference stress for use in stress display
28. Y_{zz}, reference stress for use in stress display
29. Y_{xy}, reference stress for use in stress display
30. Y_{yz}, reference stress for use in stress display
31. Y_{xz}, reference stress for use in stress display

The default values of each reference stress is 1.0.

For an orthotropic material with the 1-, 2-, and 3-directions aligned with the x-, y-, and z-directions respectively, the flexibility matrix components in terms of the engineering constants are

$$[a_{ij}] = \begin{bmatrix} \frac{1}{E_1} & -\frac{\nu_{21}}{E_2} & -\frac{\nu_{31}}{E_3} & 0 & 0 & 0 \\ -\frac{\nu_{12}}{E_1} & \frac{1}{E_2} & -\frac{\nu_{32}}{E_3} & 0 & 0 & 0 \\ -\frac{\nu_{13}}{E_1} & -\frac{\nu_{23}}{E_2} & \frac{1}{E_3} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{G_{23}} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{G_{13}} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{G_{12}} \end{bmatrix}$$

PROP BTAB 2 21 (concluded)

where

$E_1,\ E_2,\ E_3$ = Young's moduli in 1, 2, and 3 directions, respectively.

ν_{ij} = Poisson's ratio for transverse strain in the j-direction when stressed in the i-direction.

$G_{23},\ G_{13},\ G_{12}$ = shear moduli in the 2-3, 1-3, and 1-2 planes, respectively.

$$\frac{\nu_{ij}}{E_i} = \frac{\nu_{ji}}{E_j} \qquad i,j = 1,2,3$$

Thus, there are three reciprocal relations that must be satisfied for an orthotropic material. Moreover, only ν_{12}, ν_{13}, and ν_{23} need be further considered since ν_{21}, ν_{31}, and ν_{32} can be expressed in terms of the first-mentioned Poisson's ratios and the Young's moduli.

QJJT BTAB 2 9

Created in processor TAB.
NJ = Number of Joints
NI = 9
Type = real

Contents of each entry:

1. a_{11}
2. a_{21}
3. a_{31}
4. a_{12}
5. a_{22}
6. a_{32}
7. a_{13}
8. a_{23}
9. a_{33}

Formula:

Each entry contains a 3 × 3 matrix to convert global reference frame to alternate reference frame for that joint.

$$\begin{Bmatrix} X_a \\ Y_a \\ Z_a \end{Bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{Bmatrix} X_g \\ Y_g \\ Z_g \end{Bmatrix}$$

coordinates in alternate reference frame

coordinates in global reference frame

SA BTAB 2 13

Created from shell section properties in processor TAB.
NJ = Number of entries
Type = real
Contents vary according to section type:
For MEMBRANE, PLATE, ISOTROPIC or UNCOUPLED section types NI = 43

Contents of each entry:

1. Number indicating section type
   1 MEMBER
   2 PLATE
   3 ISOTROPIC or UNCOUPLED
2. Pointer to entry of NMAT containing material constants
3. Structural weight/area
4. d_{11}
5. d_{12}
6. d_{22}
7. d_{13}
8. d_{23}
9. d_{33}
10. d_{44}
11. d_{45}
12. d_{55}
13. d_{46}
14. d_{56}
15. d_{66}

(4–15: flexibility coefficients)

16.
17.
18.
19.
20.
21.
22.
23.
24.
25.

Not used.

26. f_{11}
27. f_{21}
28. f_{31}
29. f_{41}
30. f_{51}
31. f_{61}
32. f_{12}
33. f_{22}
34. f_{32}
35. f_{42}
36. f_{52}
37. f_{62}
38. f_{13}
39. f_{23}
40. f_{33}
41. f_{43}
42. f_{53}
43. f_{63}

(26–43: stress coefficients)

SA BTAB 2 13 (continued)

For COUPLED section types:
NI = 43

Contents of each entry:

1. Number indicating section type
   4 = COUPLED
2. Pointer to entry of NMAT containing material constants
3. Structural weight/area
4. d_{11}
5. d_{12}
6. d_{22}
7. d_{13}
8. d_{23}
9. d_{33}
10. d_{14}
11. d_{24}
12. d_{34}
13. d_{44}
14. d_{15}
15. d_{25}
16. d_{35}
17. d_{45}
18. d_{55}
19. d_{16}
20. d_{26}
21. d_{36}
22. d_{46}
23. d_{56}
24. d_{66}

(4. – 24.: flexibility coefficients)

25. Number of layers
26. f_{11}
27. f_{21}
28. f_{31}
29. f_{41}
30. f_{51}
31. f_{61}
32. f_{12}
33. f_{22}
34. f_{32}
35. f_{42}
36. f_{52}
37. f_{62}
38. f_{13}
39. f_{23}
40. f_{33}
41. f_{43}
42. f_{53}
43. f_{63}

(26. – 43.: stress coefficients)

SA BTAB 2 13 (concluded)

For LAMINATE section types: NI = 25 + (18 times number of layers)

Contents of each entry:

1. Number indicating section type
   5 = LAMINATE
2. Pointer to entry of MATC containing material constants
3. Structural weight/area
4. d_{11}
5. d_{12}
6. d_{22}
7. d_{13}
8. d_{23}
9. d_{33}
10. d_{14}
11. d_{24}
12. d_{34}
13. d_{44}
14. d_{15}
15. d_{25}
16. d_{35}
17. d_{45}
18. d_{55}
19. d_{16}
20. d_{26}
21. d_{36}
22. d_{46}
23. d_{56}
24. d_{66}

(items 4–24: flexibility coefficients)

25. Number of layers
26. g^1_{11}
    .
    .
    .
43. g^1_{36}

(items 26–43: stress recovery coefficients for first layer)

44. g^2_{11}
    .
    .
    .
61. g^2_{36}

(items 44–61: stress recovery coefficients for second layer)

62. → (25 + 18 * number of layers)
    Eighteen additional values for each successive layer.

SB BTAB 2 14

Created from subprocessor SB in processor TAB.

NJ = Number of entries
NI = 4
Type = real

Contents of each entry:

1. Thickness of E44 element
2. }
3. } Not used.
4. }

STAT DISP iset ncon

iset = Load set
ncon = Constraint case
Created in processor SSOL
SYSVEC format. See "APPL FORC iset".

Contents:

Each entry contains static displacements for that joint in each active direction.

STAT REAC iset ncon

iset = Load set
ncon = Constraint case
Created in processor SSOL
SYSVEC format. See "APPL FORC iset".

Contents:

Each entry contains static reactions for that joint in each active direction.

STRS E21 iset icase

iset = Load set
icase = Load case within set
Created in processor GSF.
NJ = Number of E21 elements
NI = 52
Type = real

The dataset contains NJ nominal records, NI items per record.

Contents of each record:

1. Group number
2. Element number within group
3. Joint #1
4. Joint #2
5. Max. combined P/A + bending (tension)
6. Max. combined P/A + bending (compression)
7. P/A
8. Transverse shear stress, S
9. Transverse shear stress, S
10. Twist shear
11. Shear force, end 1, direction 1
12. Shear force, end 1, direction 2
13. Axial force, end 1, direction 3
14. Moment, end 1, direction 4
15. Moment, end 1, direction 5
16. Moment, end 1, direction 6
17. Shear force, end 2, direction 1
18. Shear force, end 2, direction 2
19. Axial force, end 2, direction 4
20. Moment, end 2, direction 4
21. Moment, end 2, direction 5
22. Moment, end 2, direction 6
23. Not used
24. I_1
25. α_1
26. I_2
27. α_2
28. Area
29. f_1
30. f_2
31. z_1
32. z_2
33. Θ
34. q_1
35. q_2
36. q_3
37. NY = number of points for stress
38. y_{11}
39. y_{12}
40. y_{21}
41. y_{22}
42. y_{31}
43. y_{32}
44. y_{41}
45. y_{42}
46. b_1
47. t_1
48. b_2
49. t_2
50. b_3
51. t_3

STRS E22 iset icase

iset = Load set
icase = Load case within set
Created in processor GSF
NJ = Number of E22 elements
NI = 16
Type = real

The dataset contains NJ nominal records, NI items per record.

Contents of each record:

1. Group number
2. Element number within group
3. Joint #1
4. Joint #2
5. Force in direction 1 at joint 1
6. Force in direction 2 at joint 1
7. Force in direction 3 at joint 1
8. Moment about axis 1 at joint 1
9. Moment about axis 2 at joint 1
10. Moment about axis 3 at joint 1
11. Force in direction 1 at joint 2
12. Force in direction 2 at joint 2
13. Force in direction 3 at joint 2
14. Moment about axis 1 at joint 2
15. Moment about axis 2 at joint 2
16. Moment about axis 3 at joint 2

STRS E23 iset icase

iset = Load set
icase = Load case within set
Created in processor GSF
NJ = Number of E23 elements
NI = 6
Type = real

The dataset contains NJ nominal records, NI items per record.

Contents of each record:

1. Group number
2. Element number within group
3. Joint #1
4. Joint #2
5. Force in element
6. Stress in element

STRS E24 iset icase

iset = Load set
icase = Load case within set
Created in processor GSF
NJ = Number of E24 elements
NI = 18
Type = real

The dataset contains NJ nominal records, NI items per record.

Contents of each record:

1. Group number
2. Element number within group
3. Joint #1
4. Joint #2
5. Axial force at joint 1
6. Transverse shear at joint 1
7. Moment at joint 1
8. Axial force at joint 2
9. Transverse shear at joint 2
10. Moment at joint 2
11. Axial stress at joint 1
12. Shear stress at joint 1
13. Bending stress on upper surface at joint 1
14. Bending stress on lower surface at joint 1
15. Axial stress at joint 2
16. Shear stress at joint 2
17. Bending stress on upper surface at joint 2
18. Bending stress on lower surface at joint 2

STRS E25 iset icase

iset = Load set
icase = Load case within set
Created in processor GSF.
NJ = Number of E25 elements
NI = 16
Type = real

The dataset contains NJ nominal records, NI items per record.

Contents of each record:

1. Group number
2. Element number within group
3. Joint #1
4. Joint #2
5. Force in direction 1 at joint 1
6. Force in direction 2 at joint 1
7. Force in direction 3 at joint 1
8. Moment about axis 1 at joint 1
9. Moment about axis 2 at joint 1
10. Moment about axis 3 at joint 1
11. Force in direction 1 at joint 2
12. Force in direction 2 at joint 2
13. Force in direction 3 at joint 2
14. Moment about axis 1 at joint 2
15. Moment about axis 2 at joint 2
16. Moment about axis 3 at joint 2

STRS E31 iset icase

Contains stress resultants calculated in the element reference frame.

iset Load set
icase Load case within set
Created in processor GSF.
NJ Number of E31 elements
NI 11
Type real

The dataset contains NJ nominal records, NI items per record.

Contents of each record:

1. Group number
2. Element number within group
3. Joint #1
4. Joint #2
5. Joint #3
6. T_{11}
7. T_{22}
8. T_{22}
9. Tractive force in x-direction N_x
10. Tractive force in y-direction N_y
11. Shearing force N_{xy}

Formulae:

S_x N_x/thickness
S_y N_y/thickness
T_{xy} N_{xy}/thickness

STRS E32 iset icase

Contains stress resultants calculated in the element reference frame.

iset = Load set
icase = Load case within set
Created in processor GSF.
NJ = Number of E32 elements
NI = 28
Type = real

The dataset contains NJ nominal records, NI items per record.

Contents of each record:

1. Group number
2. Element number within group
3. Joint #1
4. Joint #2
5. Joint #3
6. Not used
7. Index of section property dataset entry for element section properties
8. Section type code
9. M_x Bending moment about-axis at joint 1
10. M_y Bending moment about y-axis at joint 1
11. M_{xy} Twisting moment at joint 1
12. Q_x Transverse shear in x-direction at joint 1
13. Q_y Transverse shear in y-direction at joint 1
14. M_x Bending moment about x-axis at joint 2
15. M_y Bending moment about y-axis at joint 2
16. M_{xy} Twisting moment at joint 2
17. Q_x Transverse shear in x-direction at joint 2
18. Q_y Transverse shear in y-direction at joint 2
19. M_x Bending moment about x-axis at joint 3
20. M_y Bending moment about y-axis at joint 3
21. M_{xy} Twisting moment at joint 3
22. Q_x Transverse shear in x-direction at joint 3
23. Q_y Transverse shear in y-direction at joint 3

STRS E32 iset icase (concluded)

24. M_x Bending moment about x-axis at the center
25. M_y Bending moment about y-axis at the center
26. M_{xy} Twisting moment at the center
27. Q_x Transverse shear in x-direction at the center
28. Q_y Transverse shear in y-direction at the center

Formulae:

$$S_x = f_{4j} M_x \qquad f_{ij} = 1/\text{thickness for } i \text{ and } j = 1,2,3$$
$$S_y = f_{5j} M_y \qquad f_{42} = f_{52} = -f_{62} = 6/(\text{thickness})^2$$
$$T_{xy} = f_{6j} M_{xy} \qquad f_{43} = f_{53} = -f_{63} = -6/(\text{thickness})^2$$

STRS E33 iset icase

Contains stress resultants calculated in the element reference frame.

iset = Load set
icase = Load case within set
Created in processor GSF.
NJ = Number of E33 elements
NI = 31
Type = real

The dataset contains NJ nominal records, NI items per record.

Contents of each record:

1. Group number
2. Element number within group
3. Joint #1
4. Joint #2
5. Joint #3
6. Not used
7. Index of section property dataset entry for element section properties
8. Section type code
9. N_x Tractive force in x-direction
10. N_y Tractive force in y-direction
11. N_{xy} Shearing force
12. M_x Bending moment about x-axis at joint 1
13. M_y Bending moment about y-axis at joint 1
14. M_{xy} Twisting moment at joint 1
15. Q_x Transverse shear in x-direction at joint 1
16. Q_y Transverse shear in y-direction at joint 1
17. M_x Bending moment about x-axis at joint 2
18. M_y Bending moment about y-axis at joint 2
19. M_{xy} Twisting moment at joint 2
20. Q_x Transverse shear in x-direction at joint 2
21. Q_y Transverse shear in y-direction at joint 2
22. M_x Bending moment about x-axis at joint 3
23. M_y Bending moment about y-axis at joint 3

STRS E33 iset icase (concluded)

24. M_{xy} Twisting moment at joint 3
25. Q_x Transverse shear in x-direction at joint 3
26. Q_y Transverse shear in y-direction at joint 3
27. M_x Bending moment about x-axis at the center
28. M_y Bending moment about y-axis at the center
29. M_{xy} Twisting moment at the center
30. Q_x Transverse shear in x-direction at the center
31. Q_y Transverse shear in y-direction at the center

Formulae:

$$S_x = f_{1j}N_x + f_{4j}M_x \qquad f_{ij} = 1/\text{thickness for } i \text{ and } j = 1,2,3$$
$$S_y = f_{2j}N_y + f_{5j}M_y \qquad f_{42} = f_{52} = -f_{62} = 6/(\text{thickness})^2$$
$$T_{xy} = f_{3j}N_{xy} + f_{6j}M_{xy} \qquad f_{43} = f_{53} = -f_{63} = -6/(\text{thickness})^2$$

STRS E41 iset icase

Contains stress resultants calculated in the element reference frame.

iset = Load set
icase = Load case within set
Created in processor GSF.
NJ = Number of E41 elements
NI = 23
Type = real

The dataset contains NJ nominal records, NI items per record.

Contents of each record:

1. Group number
2. Element number within group
3. Joint #1
4. Joint #2
5. Joint #3
6. Joint #4
7. Index of section property dataset entry for element section properties
8. Section type code
9. N_x Tractive force in x-direction at joint 1
10. N_y Tractive force in y-direction at joint 1
11. N_{xy} Shearing force at joint 1
12. N_x Tractive force in x-direction at joint 2
13. N_y Tractive force in y-direction at joint 2
14. N_{xy} Shearing force at joint 2
15. N_x Tractive force in x-direction at joint 3
16. N_y Tractive force in y-direction at joint 3
17. N_{xy} Shearing force at joint 3
18. N_x Tractive force in x-direction at joint 4
19. N_y Tractive force in y-direction at joint 4
20. N_{xy} Shearing force at joint 4
21. N_x Tractive force in x-direction at the center
22. N_y Tractive force in y-direction at the center
23. N_{xy} Shearing force at the center

STRS E41 iset icase (concluded)

Formulae:

$$S_x = N_x/\text{thickness}$$
$$S_y = N_y/\text{thickness}$$
$$T_{xy} = N_{xy}/\text{thickness}$$

STRS E42 iset icase

Contains stress resultants calculated in the element reference frame.

iset = Load set
icase = Load case within set
Created in processor GSF.
NJ = Number of E42 elements
NI = 33
Type = real

The dataset contains NJ nominal records, NI items per record.

Contents of each record:

1. Group number
2. Element number within group
3. Joint #1
4. Joint #2
5. Joint #3
6. Joint #4
7. Index of section property dataset entry for element section properties
8. Section type code
9. M_x Bending moment about x-axis at joint 1
10. M_y Bending moment about y-axis at joint 1
11. M_{xy} Twisting moment at joint 1
12. Q_x Transverse shear in x-direction at joint 1
13. Q_y Transverse shear in y-direction at joint 1
14. M_x Bending moment about x-axis at joint 2
15. M_y Bending moment about y-axis at joint 2
16. M_{xy} Twisting moment at joint 2
17. Q_x Transverse shear in x-direction at joint 2
18. Q_y Transverse shear in y-direction at joint 2
19. M_x Bending moment about x-axis at joint 3
20. M_y Bending moment about y-axis at joint 3
21. M_{xy} Twisting moment at joint 3
22. Q_x Transverse shear in x-direction at joint 3
23. Q_y Transverse shear in y-direction at joint 3

STRS E42 iset icase (concluded)

24. M_x Bending moment about x-axis at joint 4
25. M_y Bending moment about y-axis at joint 4
26. M_{xy} Twisting moment at joint 4
27. Q_x Transverse shear in x-direction at joint 4
28. Q_y Transverse shear in y-direction at joint 4
29. M_x Bending moment about x-axis at the center
30. M_y Bending moment about y-axis at the center
31. M_{xy} Twisting moment at the center
32. Q_x Transverse shear in x-direction at the center
33. Q_y Transverse shear in y-direction at the center

Formulae:

$$S_x = f_{4j} M_x \qquad f_{42} = f_{52} = -f_{62} = -6/(\text{thickness})^2$$
$$S_y = f_{5j} M_y \qquad f_{43} = f_{53} = -f_{63} = -6/(\text{thickness})^2$$
$$T_{xy} = f_{6j} M_{xy}$$

STRS E43 iset icase

Contains stress resultants calculated in the element reference frame.

iset = Load set
icase = Load case within set
Created in processor GSF
NJ = Number of E43 elements
NI = 48
Type = real

The dataset contains NJ nominal records, NI items per record.

Contents of each record:

1. Group number
2. Element number within group
3. Joint #1
4. Joint #2
5. Joint #3
6. Joint #4
7. Index of section property dataset entry for element section properties
8. Section type code
9. N_x Tractive force in x-direction at joint 1
10. N_y Tractive force in y-direction at joint 1
11. N_{xy} Shearing force at joint 1
12. N_x Tractive force in x-direction at joint 2
13. N_y Tractive force in y-direction at joint 2
14. N_{xy} Shearing force at joint 2
15. N_x Tractive force in x-direction at joint 3
16. N_y Tractive force in y-direction at joint 3
17. N_{xy} Shearing force at joint 3
18. N_x Tractive force in x-direction at joint 4
19. N_y Tractive force in y-direction at joint 4
20. N_{xy} Shearing force at joint 4
21. N_x Tractive force in x-direction at the center
22. N_y Tractive force in y-direction at the center
23. N_{xy} Shearing force at the center

STRS E43 iset icase (concluded)

24. M_x Bending moment about x-axis at joint 1
25. M_y Bending moment about y-axis at joint 1
26. M_{xy} Twisting moment at joint 1
27. Q_x Transverse shear in x-direction at joint 1
28. Q_y Transverse shear in y-direction at joint 1
29. M_x Bending moment about x-axis at joint 2
30. M_y Bending moment about y-axis at joint 2
31. M_{xy} Twisting moment at joint 2
32. Q_x Transverse shear in x-direction at joint 2
33. Q_y Transverse shear in y-direction at joint 2
34. M_x Bending moment about x-axis at joint 3
35. M_x Bending moment about y-axis at joint 3
36. M_{xy} Twisting moment at joint 3
37. Q_x Transverse shear in x-direction at joint 3
38. Q_y Transverse shear in y-direction at joint 3
39. M_x Bending moment about x-axis at joint 4
40. M_y Bending moment about y-axis at joint 4
41. M_{xy} Twisting moment at joint 4
42. Q_x Transverse shear in x-direction at joint 4
43. Q_y Transverse shear in y-direction at joint 4
44. M_x Bending moment about x-axis at the center
45. M_y Bending moment about y-axis at the center
46. M_{xy} Twisting moment at the center
47. Q_x Transverse shear in x-direction at the center
48. Q_y Transverse shear in y-direction at the center

Formulae:

$$S_x = f_{1j}N_x + f_{4j}M_x \qquad f_{ij} = 1/\text{thickness for } i \text{ and } j = 1,2,3$$
$$S_y = f_{2j}N_y + f_{5j}M_y \qquad f_{42} = f_{52} = -f_{62} = 6/(\text{thickness})^2$$
$$T_{xy} = f_{3j}N_{xy} + f_{6j}M_{xy} \qquad f_{43} = f_{53} = -f_{63} = 6/(\text{thickness})^2$$

STRS E44 iset icase

iset = Load set
icase = Load case within set
Created in processor GSF.
NJ = Number of E44 elements
NI = 8
Type = real

The dataset contains NJ nominal records, NI items per record.

Contents of each record:

1. Group number
2. Element number within group
3. Joint #1
4. Joint #2
5. Joint #3
6. Joint #4
7. Element thickness
8. Shear stress

TEMP Exx iset icase

Exx = Element name
iset = Load set
icase = Load case within Load set
Created in processor AUS.
NJ = Number of elements of this type.
Type = real

For 2-node elements (Not defined for E25 elements):
NI = 3
Contents of each entry:

1. Average temperature of the element
2. Transverse gradient in direction 1
3. Transverse gradient in direction 2

For 3-node elements (Not defined for E32 elements):
NI = 3
Contents of each entry:

1. Temperature at joint 1 of element
2. Temperature at joint 2 of element
3. Temperature at joint 3 of element

For 4-node elements (Not defined for E42 elements):
NI = 4
Contents of each entry:

1. Temperature at joint 1 of element
2. Temperature at joint 2 of element
3. Temperature at joint 3 of element
4. Temperature at joint 4 of element

Formula:

Total effective temperature at node n	=	Element temperature at node n	+	**Nodal temperature from block "icase" of dataset "NODA TEMP iset"**

TEXT BTAB 2 1

Created from TEXT card(s) in processor TAB.
Type = alphanumeric

Contains data in text.

VIBR EVAL iset ncon

iset = Load set
ncon = Constraint case
Created in processor EIG.
NJ = 1
NI = Number of eigenvalues
Type = real

Contains eigenvalues corresponding to each eigenvector in "VIBR MODE".

VIBR MODE iset ncon

iset = Load set
ncon = Constraint case
Created in processor EIG.
SYSVEC format. See "APPL FORC iset".

Contents:

Each block of data contains one eigenvector corresponding to an eigenvalue stored in "VIBR EVAL". Data is stored for each joint in each active direction.

APPENDIX B. Instructions for Installation of the Testbed on VAX/VMS

1) Create top level directory [NICESPAR] on the disk where the software is to be installed; the total disk space required for installation is 15000 disk blocks.

2) Set the default device to the name of the device used in step 1. Mount the delivery tape and enter the following commands to unload the tape:

```
$ ALLOCATE "tape_device_name" TAPE
$ MOUNT/FOREIGN TAPE:
$ BACKUP/LOG TAPE:NICESPAR.BCK/SELECT=[NICESPAR...]  [NICESPAR...]
$ DISMOUNT TAPE:
$ DEALLOCATE TAPE:
```

3) Make the following system wide logical name definition for the NICESPAR root directory in the system startup procedure or in user login procedures:

```
$ DEFINE NS$ROOT  "dev":[NICESPAR]
```

where "dev" is the name of the disk that the files reside on.

4) NICESPAR users must include in their login procedures the following command so that NICESPAR symbols and logical names are defined for their process:

```
$ @NS$ROOT:LOGIN.COM
```

The executable file for NICESPAR created under VMS 4.3 is included on the delivery tape and can be executed at this time.

NOTE: Steps 1-3 must be performed by a user with privileges to create top level directories and define system wide logical names.

The directories which are created by the installation process are:

Logical Name	Directory Name	Contents
NS$ROOT	[NICESPAR]	LOGIN.COM, SUBMIT.COM
NS$MSC	[NICESPAR.MSC]	Master source files and maintenance procedure files for NICE/SPAR
NS$SRC	[NICESPAR.SRC]	Fortran source files
NS$LIS	[NICESPAR.LIS]	Fortran compiler output
NS$OBJ	[NICESPAR.OBJ]	NICE/SPAR object libraries
NS$EXE	[NICESPAR.EXE]	NICE/SPAR executable and map
NS$DEMO	[NICESPAR.DEMO]	Demonstration procedure files
NICE$OLB	[NICESPAR.NICE.OLB]	NICE object library
NICE$MSC	[NICESPAR.NICE.MSC]	NICE master source files
NICE$EXE	[NICESPAR.NICE.EXE]	NICE utility executable files

If a new executable version of NICE/SPAR must be created from the source version, enter the following commands:

```
$ SET DEFAULT NS$MSC
$ @MAKENS
```

The procedures MAKENS.COM, MAKESPARPROC.COM, AND LINKNS.COM, which reside in the directory NS$MSC, are listed below.

MAKENS.COM procedure listing:

```
$!  makens.com
$!
$!  Procedure to extract NICE/SPAR VAX source version from master source
$!        files and link the executable version in NS$EXE:NICESPAR.EXE
$!
$ set def ns$msc:
$ on error then $goto end
$ max/for/vax/sic/wc <sparinc.ams >/inc NICE DOUBLE
$ @makesparproc nicespar
$ @makesparproc aus
$ @makesparproc dcu
$ @makesparproc dr
$ @makesparproc e
$ @makesparproc eig
$ @makesparproc eks
$ @makesparproc eld
$ @makesparproc enl
$ @makesparproc eqnf
$ @makesparproc gsf
$ @makesparproc imp
$ @makesparproc inv
$ @makesparproc k
$ @makesparproc kg
$ @makesparproc lau
$ @makesparproc m
$ @makesparproc pama
$ @makesparproc pkma
$ @makesparproc ps
$ @makesparproc plta
$ @makesparproc pltb
$ @makesparproc spargraf
$ @makesparproc prte
$ @makesparproc psf
$ @makesparproc ssol
$ @makesparproc tab
$ @makesparproc topo
$ @makesparproc vec
$ @makesparproc vprt
$ @makesparproc csm1
$ @makesparproc rseq
$ @makesparproc shel
```

```
$ @makesparproc shell          !  Library of Lockheed shell subroutines
$ @makesparproc tgeo
$ @makesparproc ssta
$ @makesparproc trta
$ @makesparproc trtb
$ @makesparproc trtg
$ @makesparproc tafp
$ @makesparproc mtp
$ @makesparproc view
$ @makesparproc tak
$ @makesparproc tads
$ @makesparproc thermlib1      !  Library of thermal routines
$ @makesparproc thermlib2      !  Library of thermal routines
$ @makesparproc crutil         !  Library of Lockheed corotational routines
$ @makesparproc gsutil         !  Library of Lockheed utility routines
$ @makesparproc nsutil         !  Library of Lockheed utility routines
$ @makesparproc nsparlib1
$ @makesparproc nsparlib2
$ @makesparproc nsparlib3
$ lib/cre ns$obj:nsparlib.olb ns$obj:nsparlib1,nsparlib2,nsparlib3
$ del ns$obj:*.obj;*
$ @linkns
$ end:
$ exit
```

MAKESPARPROC.COM Procedure Listing:

```
$ IF P1.EQS."" THEN $INQUIRE P1 "Enter processor name"
$ IF P1.EQS."" THEN $EXIT
$ on error then goto end
$ @checkdate ns$msc:'p1'.ams ns$obj:'p1".olb
$ if done .eq.  0 then $goto compile
$ write sys$output p1," is up-to-date"
$ goto end
$ compile:
$!  Extract VMS/VAX version of SPAR processor
$ INCLUDE <NS$MSC:'P1'.AMS >NS$MSC:'P1'.IMS
$!  set up keys for MAX; include TEK key only if PLOT10LIB is defined
$ keys = "DOUBLE NICE"
$ if plot10lib .nes.  "" then $ keys = "DOUBLE NICE TEK"
$ MAX/FOR/VAX/WC/UC/SIC/L <ns$msc:'P1'.IMS >ns$src:'p1'.FOR 'keys'
$ DEL NS$MSC:'P1'.IMS;*
$!  Compile extracted code and make user library
$ FOR/LIS=NS$LIS:/OBJ=NS$OBJ: NS$SRC:'P1.FOR
$ LIB/CRE NS$OBJ:'P1' NS$OBJ:'P1'
$ del ns$src:'p1'.*;*
$ end:
$ pur ns$lis:'p1'.*,ns$obj:'p1'.*
$ exit
```

LINKNS.COM Procedure Listing:

```
$!  linkns.com - Link NICE/SPAR executable
$ if plot10lib .nes.  "" then $goto link
$   if f$search("ns$obj:plot10.olb") .eqs.  "" then -
       lib/cre ns$obj:plot10.olb
$   plot10lib :== ns$obj:plot10/l
$ link:
$ link/exec=ns$exe:/map=ns$exe:  -
ns$obj:nicespar/l/include=(nicespar,comdata), -
aus/l, -
csm1/l, -
dcu/l, -
dr/l, -
e/l, -
eig/l, -
eks/l, -
eld/l, -
eqnf/l, -
gsf/l, -
inv/l, -
k/l, -
kg/l, -
m/l, -
plta/l, -
pltb/l, -
psf/l, -
rseq/l, -
ssol/l, -
tab/l, -
topo/l, -
vprt/l, -
ps/l, -
pama/l, -
pkma/l, -
prte/l, -
lau/l, -
enl/l, -
vec/l, -
shel/l, -
mtp/l, -
ssta/l, -
tafp/l, -
tgeo/l, -
trta/l, -
trtb/l, -
trtg/l, -
tak/l, -
tads/l, -
view/l, -
nsparlib/l,spargraf/l, -                ! NICE/SPAR libraries
thermlib2/l, -                          ! THERMAL lib split (2/87)
thermlib1/l, -                          ! THERMAL libraries (added 12/86)
```

```
crutil.olb/l, -                          !  Corotational routines
shell.olb/l, -                           !  Lockheed shell routines
nsutil.olb/l,gsutil.olb/l, -             !  Lockheed utilities
'plot10lib', -                           !  PLOT10 library
'nicelib'/l                              !  NICE library
$ pur ns$exe:nicespar.*
$ exit
```

APPENDIX C. Descriptions of New Testbed Processors

NICE/SPAR processors have been developed by the CSM group to perform functions required for the analysis of some problems of current interest. Usage descriptions of the following installed processors are presented in this appendix:

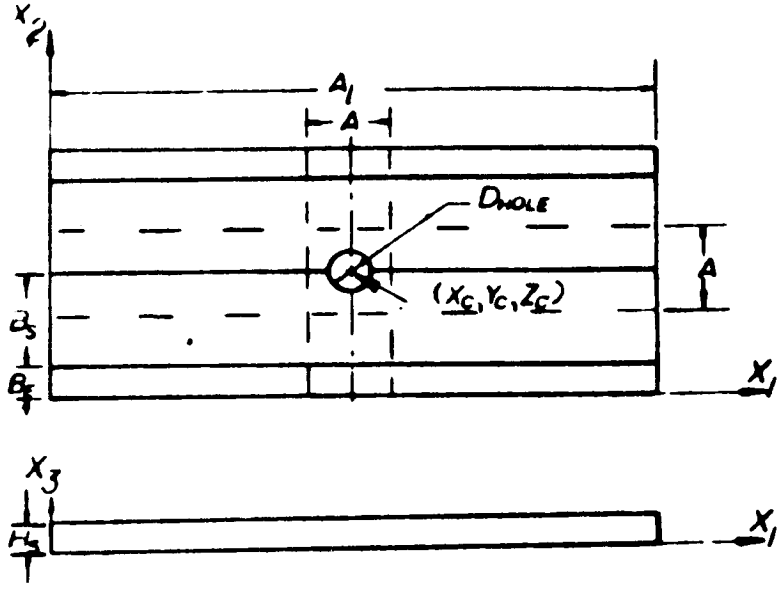
CSM1 - Focus Problem Data Generator
LAU - Laminate Analysis Utility
RSEQ - Joint Elimination Sequence Generation

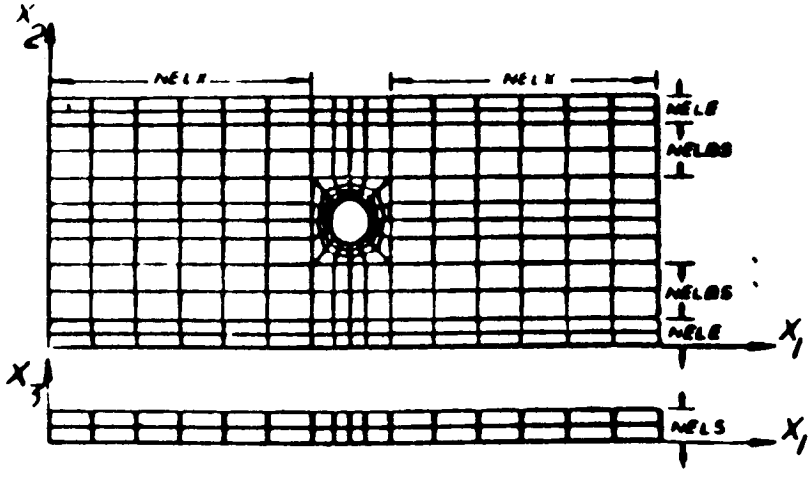
Documentation for installed processors ENL and VEC is given in Reference 9.

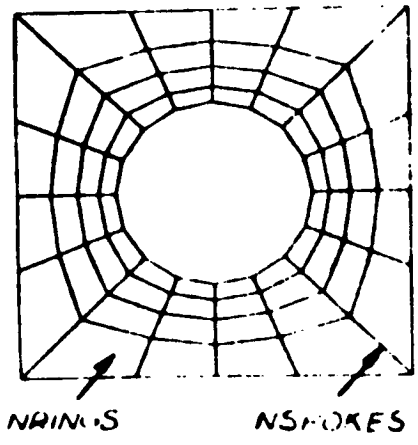
CSM1 – Focus Problem Data Generator

Processor CSM1 is used to model a rectangular, blade-stiffened panel with a single, centered hole. CSM1 generates nodal coordinates, element connectivities, boundary conditions, and applied displacements in the form of NICE/SPAR procedures written to file PANEL.PRC (in the user's default directory).

As shown below, the geometry of the panel is defined by parameters A, A_l, d_{hole}, b_e, b_s, and h_s. The finite element grid is defined by the parameters NELX, NELE, NELBS, NSPOKES, NRINGS, and NELS.







REQUIRED INPUT:

User input for CSM1 is in the form of two TABLEs formed in AUS. The input parameters* are defined as follows:

Integer parameters:

NNPE:	Number of nodes (joints) per elements (3, 4, or 9)
IOPT:	Element option: 0 for E33 or E43 elements 1-7 for experimental element options 1-7
NRINGS:	Number of rings *of elements* (4 or 9 node) around hole
NSPOKES:	Number of radial spokes *of nodes* normal to hole boundary (must be a multiple of 8)
IWALL:	Panel section property flag (NSECT for panel skin)
JWALL:	Stiffener section property flag (NSECT for panel stiffeners)
IREF:	Material reference frame for panel skin
JREF:	Material reference frame for panel stiffeners
NELX:	Number of elements (4 or 9 node) between $0. < x < \frac{(A_l - A)}{2}$
NELE:	Number of elements (4 or 9 node) from panel edge to outside stiffener; between $0.0 < y < b_e$
NELBS:	Number of elements (4 or 9 node) between interior stiffeners; between $b_s < y < [b_e + b_s - \frac{A}{2}]$
NELS:	Number of elements (4 or 9 node) across height of stiffener; if nels=0, there are no stiffeners
IFILL:	Flag to fill in hole: 0=No fill-in, 1=Fill-in

Floating point parameters:

A:	Length of a side of the central square region; A > DHOLE
AL:	Overall length of panel
BE:	Distance between panel edge and outside stiffener
BS:	Distance between interior stiffeners
DHOLE:	Diameter of hole
HS:	Height of stiffener
RAT:	Mesh grading factor - near zero gives equal spacing of rings; as RAT $\Rightarrow$ 1.0, finer mesh near hole
XC,YC,ZC:	Local coordinates of center of hole

The eight elements will be generated in groups defined as follows:

GROUP 1	Elements within the A by A square around the hole. Does NOT include elements formed from filling in the hole.

* If three node elements are to be used, input parameters are those of the comparable 4-node element problem. CSM1 generates the three node element grid by dividing each 4 node element into two three node elements.

GROUP 2 Portion of center stiffener within A by A square around hole
GROUP 3 Panel skin
GROUP 4 Stiffener at $y = b_e$
GROUP 5 $0.0 < x < \frac{A_l - A}{2}$ portion of center stiffener
GROUP 6 $\frac{A_l + A}{2} < x < A_l$ portion of center stiffener
GROUP 7 Stiffener at $y = b_e + 2 * b_s$
GROUP 8 Elements formed by filling in hole

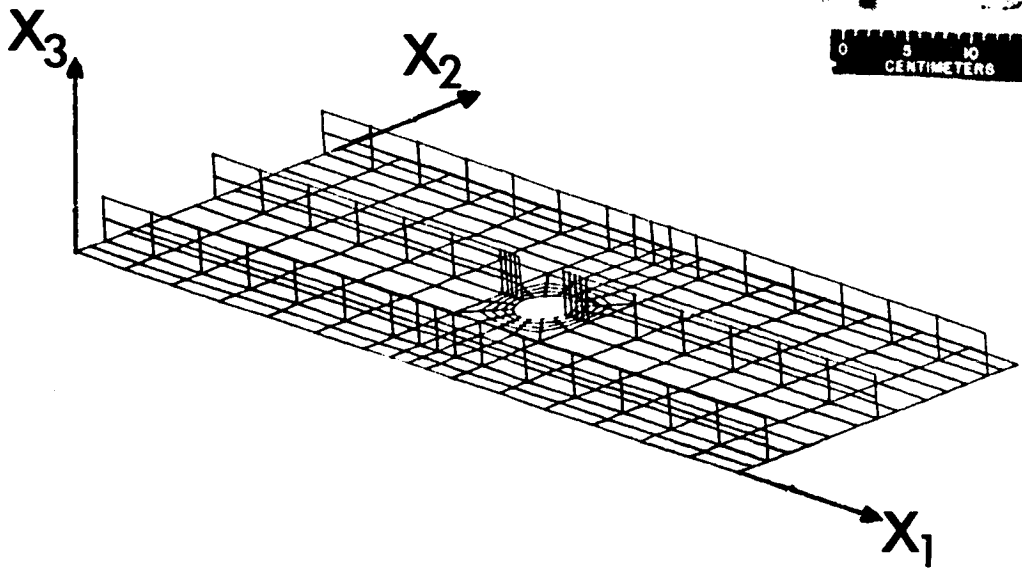
If a four-node element mesh is being generated, checks are done within CSM1 to verify the compatibility of the mesh with a 9-node element mesh. If the mesh is compatible then execution proceeds as normal with no extra print out. If the mesh is not compatible, then a warning is printed to file PANEL.PRC giving the name of the offending parameter. The warning, if it is present, will be the *FIRST* thing written to PANEL.PRC. Execution will continue in spite of the warning. In general, if compatible grids are desired, NELS, NELX, NELE, NELBS, NSPOKES, and NRINGS should be even. If compatible grids are unimportant, simply ignore the warning as it will be considered a comment by NICE/SPAR.

A full explanation of user input is best done by example. In the example provided, all input data is supplied by the user via the procedure MESH_FOCUS, which is called by procedure MAIN prior to the execution of CSM1. The Focus Problem along with a finite element model are shown in the following figures:

BLADE-STIFFENED GRAPHITE-EPOXY PANEL WITH A DISCONTINUOUS STIFFENER

– FOCUS PROBLEM –

- GRAPHITE-EPOXY (T300/5208)
- FLAT PANEL WITH THREE BLADE STIFFENERS
- 30 IN. LONG
- 11.5 IN. WIDE
- STIFFENER SPACING OF 4.5 IN.
- STIFFENER HEIGHT OF 1.4 IN.
- 2.0-IN.-DIAMETER HOLE
- 25-PLY PANEL SKIN
- 24-PLY BLADE STIFFENERS
- AXIALLY LOADED WITH LOADED ENDS CLAMPED AND SIDES FREE



MESH_FOCUS:

Procedure MESH_FOCUS, listed on the following page, contains the user input required by CSM1. Two TABLEs have been formed, CSMP FOCS 1 1 and CSMP FOCS 1 2. These two TABLEs must be formed and their contents must be listed in the order shown. CSMP FOCS 1 1 contains all (and only) integer input data. The first four entries are: NNPE, IOPT, NRINGS, NSPOKES. The next 20 entries contain boundary conditions. Unfortunately, the representation of the boundary conditions is the exact opposite of the usual representation in NICE/SPAR. Here, a

0 - indicates that the d.o.f. is *CONSTRAINED TO ZERO*
1 - indicates that the d.o.f. is *FREE*

In addition, the constraints are on the corners and edges of the panel, as indicated on the following page, and not on specific nodes. The transformation to specific nodes is made within CSM1.

The displacement is applied at the $x = 0.0$ edge and as such that edge is given a free boundary condition by the user. The x-direction displacement along this edge is set to NONZERO within CSM1 so that the uniform end shortening may be applied.

The remaining entries in this TABLE are: IWALL, JWALL, IREF, JREF, NELX, NELE, NELBS, NELS, and IFILL.

CSMP FOCS 1 2 contains all (and only) floating point input data. There are only ten entries in this TABLE: A, DHOLE, XC, YC, ZC, RAT, AL, BE, BS, and HS.

The coordinates (x_c, y_c, z_c) are local coordinates of the center of the hole. They are measured from the zero: $x_{global} = \frac{(A_l - A)}{2}$, $y_{global} = (b_e + b_s)$, $z_{global} = 0.0$.

The sixth entry in the TABLE, RAT, is a parameter which allows the user to grade the mesh near the hole. For RAT = 0., element rings will be of equal size. As RAT $\Rightarrow$ 1., the mesh becomes finer close to the hole and coarser away from the hole. RAT is only effective within the A by A square region around the hole.

```
.
.
*PROCEDURE mesh_focus
[XQT aus
.
.---  build table of integer user data
.
TABLE(NI=33,NJ=1,itype=0):  CSMP FOCS 1 1
J=1:  4  0  4  16 >    .NNPE, IOPT, NRINGS, NSPOKES
.
.    Boundary Conditions:
.     uvw ruvw
.
      110 000>       .Edge x=0.0                (Edge 1)
      111 111>       .Edge y=2*(BE+BS)          (Edge 2)
      010 000>       .Edge x=AL                 (Edge 3)
      111 111>       .Edge y=0.0                (Edge 4)
      110 000>       .Corner at (0.,0.)
      100 000>       .Corner at (0.,2*(BE+BS))
      000 000>       .Corner at (AL,2*(BE+BS))
      010 000>       .Corner at (AL,0.)
      100 000>       .Stiffeners at x=0.0
      000 000>       .Stiffeners at x=AL
.
.    IWALL JWALL IREF JREF NELX NELE NELBS NELS IFILL
.
      1     2     1    4    6    2    2    2    0
.
.---  build table of floating point user data
.
TABLE(NI=10,NJ=1):  CSMP FOCS 1 2
.
.     A    DHOLE XC   YC   ZC  RAT   AL    BE    BS   HS
.
J=1:  4.0  2.0  2.0  0.0  0.0 0.25  30.0  1.25  4.5  1.4
*END
```

CSM1:

After calling MESH_FOCUS and setting up the required TABLEs, CSM1 may be executed. Note that there are NO resets available/necessary. The processor writes the file PANEL.PRC to the user's default directory. Within PANEL.PRC, the following procedures will be found:

a) PANEL_START - Start card with correct number of joints
b) PANEL_JLOC - Joint locations in rectangular coordinates
c) PANEL_BC - Boundary conditions
d) PANEL_CONN - Element connectivity (nref,sref,etc. included)
e) PANEL_AD - Applied displacements (on $x = 0.$ edge)

All of the problem data then resides in the file PANEL.PRC. In order to use the data, the command

*ADD PANEL.PRC

must be used. *ADD redirects input to the named file which is then processed sequentially. Several .DAT files (one for each procedure contained within PANEL.PRC) are created in the user's default directory. Once PANEL.PRC has been read and the .DAT files created, control returns to the procedure, MAIN, where the individual procedures created by CSM1 may then be called. The partial runstream supplied below is an example of the use of CSM1 and its output.

```
*PROCEDURE main
[XQT tab
start 1000
*CALL mesh_focus
[XQT csm1
*ADD PANEL.PRC
[XQT tab
*CALL panel_start
jloc
*CALL panel_jloc
matc:  1 1.0 0.3
con 1
*CALL panel_bc
*CALL matdat
[XQT lau
reset key1=-1
[XQT eld
*CALL panel_conn
[XQT aus
sysvec :  appl moti
*CALL panel_ad

*END
```

LAU: A Laminate Analysis Processor for NICE/SPAR

Functions of LAU:

1. Form the coefficients for the constitutive relations for the 2-D structure being analyzed. This processor will calculate the various stiffness matrices for extension [A], bending [D], bending-extensional coupling [B], and transverse shear [CS]. For elements based on classical plate theory, a dataset is written to a NICE/SPAR library and named SA BTAB 2 13. This dataset contains the same data as generated by NICE/SPAR for "COUPLED" section properties (a total of 43 entries). For elements based on a first-order transverse shear deformation theory, two datasets are written to a NICE/SPAR library. One dataset is the previously mentioned SA BTAB 2 13 dataset and the other dataset is named PROP BTAB 2 101 which contains the elements of the matrices [A], [B], [D], and [CS] (a total of 40 entries).

2. Form the coefficients for the constitutive relations for the 3-D structure being analyzed assuming an orthotropic 3-D material. A dataset is written to the NICE/SPAR library and named PROP BTAB 2 21.

3. Calculate detailed through-the-thickness stress distributions using the available stress resultant data.

4. Evaluate various failure criteria.

NICE/SPAR LAU Input for 2-D Models:

Use the AUS processor to build tables of material data and laminate data. The following is an example of creating a material properties dataset named OMB DATA 1 1 and four laminate datasets named LAM OMB 1 1, LAM OMB 2 1, LAM OMB 3 1, and LAM OMB 4 1 where the first integer in the dataset name corresponds to the value of NSECT for that laminate.

```
[xqt aus
.
.  build table of material data
.  E11 NU12 E22 G12 G13 G23 alpha1 alpha2 wtden
.
table(ni=9,nj=4):  omb data 1 1
i=1,2,3,4,5,6,7,8,9
j=1:  19.e+6 .38 1.89e+6 .93E+6 .93E+6 .93e+6 1.e-4 1.e-4 .01
j=2:  19.e+6 .38 1.89e+6 .93E+6 .93E+6 .93e+6 1.e-4 1.e-4 .01
j=3:  1.e+7 .3 1.e+7 3.85e+6 3.85e+6 3.85e+6 1.e-4 1.e-4 .1
j=4:  18.e+6 .38 1.89e+6 .93E+6 .93E+6 .93e+6 1.e-4 1.e-4 .01
.
.  build laminate data tables
.  material type, layer thickness, angle in degrees
.
table(ni=3,nj=25,itype=0):  lam omb 1 1
.
.  skin of focus problem (NSECT=1)
.
i=1,2,3
j=1:  1 .0055 45.
j=2:  1 .0055 -45.
j=3:  1 .0055 0.
j=4:  1 .0055 0.
j=5:  1 .0055 -45.
j=6:  1 .0055 45.
j=7:  1 .0055 0.
j=8:  1 .0055 0.
j=9:  1 .0055 0.
j=10:  1 .0055 45.
j=11:  1 .0055 -45.
j=12:  1 .0055 0.
j=13:  1 .0055 0.
j=14:  1 .0055 0.
j=15:  1 .0055 -45.
j=16:  1 .0055 45.
j=17:  1 .0055 0.
j=18:  1 .0055 0.
j=19:  1 .0055 0.
j=20:  1 .0055 45.
j=21:  1 .0055 -45.
j=22:  1 .0055 0.
j=23:  1 .0055 0.
j=24:  1 .0055 -45.
```

```
j=25:  1 .0055 45.
.
table(ni=3,nj=24,itype=0):  lam omb 2 1
.
.  blade stiffeners of focus problem (NSECT=2)
.
i=1,2,3
j=1:  2 .0055 45.0
j=2:  2 .0055 -45.0
j=3:  2 .0055 0.0
j=4:  2 .0055 0.0
j=5:  2 .0055 0.0
j=6:  2 .0055 0.0
j=7:  2 .0055 0.0
j=8:  2 .0055 0.0
j=9:  2 .0055 0.0
j=10:  2 .0055 0.0
j=11:  2 .0055 0.0
j=12:  2 .0055 0.0
j=13:  2 .0055 0.0
j=14:  2 .0055 0.0
j=15:  2 .0055 0.0
j=16:  2 .0055 0.0
j=17:  2 .0055 0.0
j=18:  2 .0055 0.0
j=19:  2 .0055 0.0
j=20:  2 .0055 0.0
j=21:  2 .0055 0.0
j=22:  2 .0055 0.0
j=23:  2 .0055 -45.0
j=24:  2 .0055 45.0
.
table(ni=3,nj=1,itype=0):  lam omb 3 1
.
.  isotropic check case (NSECT=3)
.
i=1,2,3
j=1:  3 .1 0.0
.
table(ni=3,nj=16,itype=0):  lam omb 4 1
.  QUASI-ISOTROPIC MATERIAL (16 PLY SKIN) (NSECT=4)
.
.    SHELL WALL  (45/-45/0/90/45/-45/0/90)S
.
```

```
i=1,2,3
j=1:  4  0.00585  45.0
j=2:  4  0.00585 -45.0
j=3:  4  0.00585   0.0
j=4:  4  0.00585  90.0
j=5:  4  0.00585  45.0
j=6:  4  0.00585 -45.0
j=7:  4  0.00585   0.0
j=8:  4  0.00585  90.0
j=9:  4  0.00585  90.0
j=10:  4  0.00585   0.0
j=11:  4  0.00585 -45.0
j=12:  4  0.00585  45.0
j=13:  4  0.00585  90.0
j=14:  4  0.00585   0.0
j=15:  4  0.00585 -45.0
j=16:  4  0.00585  45.0
```

NICE/SPAR LAU Input for 3-D Models:

Use the AUS processor to build tables of material data and laminate data. The following is an example of creating a material properties dataset named O3D DATA 1 1 and three laminate datasets named LAM O3D 1 1, LAM O3D 2 1, and LAM O3D 3 1 where the first integer in the dataset name corresponds to the value of NSECT for that laminate.

```
[xqt aus
.
.  build table of material data
.  E11 E22 E33 G12 G23 G13 NU12 NU23 NU13 WTDEN
.      ALPHA1 ALPHA2 ALPHA3
.
table(ni=13,nj=3):  o3d data 1 1
j=1:10.e+6 10.e+6 15.e+6 3.85e+6 3.85e+6 3.85e+6 .3 .3 .3 .1>
1.e-4 1.e-4 1.e-4
j=2:10.e+6 10.e+6 25.e+6 3.85e+6 3.85e+6 3.85e+6 .3 .3 .3 .1>
1.e-4 1.e-4 1.e-4
j=3:10.e+6 10.e+6 10.e+6 3.846154e+6 3.846154e+6 3.846154e+6>
.3 .3 .3 .1 1.e-4  1.e-4  1.e-4
.
.  build laminate data tables
.  material type, layer thickness, angle in degrees
.
table(ni=3,nj=1,itype=0):  lam o3d 1 1
j=1:  2 0.0055  45.
```

```
table(ni=3,nj=2,itype=0):  lam o3d 2 1
j=1:  1 0.0055  0.
j=2:  1 0.0055  0.
table(ni=3,nj=1,itype=0):  lam o3d 3 1
j=1:  3 1.0 0.0
```

Resets for LAU:

The processor LAU has several reset parameters that can be specified on execution. These reset parameters and their default values are given as follows:

NLIB = 1
IDBG = 0
KEY1 = 1

The reset parameter NLIB is to reset which library to read and write the various material datasets. The reset parameter IDBG is to turn on additional output for debugging purposes. The reset parameter KEY1 is to reset the type of constitutive relations to be used in writing datasets to the library. If KEY1= 1, classical laminated plate theory is used wherein the effects of transverse shear stiffness is neglected. If KEY1= -1, transverse shear flexibility effects are included in the analysis provided an experimental element is used. If KEY1= -2, three dimensional constitutive relations are calculated and written to the dataset.

RSEQ – Joint Elimination Sequence Generation

Processor RSEQ supplies a joint elimination sequence by any one of four methods: Nested Dissection(fill minimizer), Minimum Degree(fill minimizer), Reverse Cuthill-McKee(profile minimizer), and Gibbs-Poole-Stockmeyer(bandwidth minimizer). The first three methods, Nested Dissection (N/D), Minimum Degree (M/D), and Reverse Cuthill-McKee (RCM), were all taken from Reference 10, while the Gibbs-Poole-Stockmeyer (GPS) algorithm was taken from BANDIT with program documentation supplied in Reference 11.

For large problems, significant savings in CPU times can usually be realized by employing one of the four joint elimination sequences. However, RSEQ is still in the experimental stages; while it has been tested (with satisfactory results) for a number of cases, there is as yet no clear indication of when one method may produce a better elimination sequence than another. Each of the four methods available work well for some, usually different, problems. It is recommended that the user make preliminary executions through RSEQ and TOPO for each of the four methods, checking the values of IC1 and IC2 printed out by TOPO. The method resulting in the smallest values of IC1 and IC2 presents the best renumbering scheme (of the four available) for the particular problem.

IC1, a cost index provided as output by TOPO, is a measure of the cost of execution of INV in terms of the number of operations required to factor the stiffness matrix; IC2, also output by TOPO, is a measure of the cost of execution of SSOL. Results for three different test problems have been included in the tables below. Table 1 lists the values of IC1 and IC2 and times spent (in CPU seconds) in INV and SSOL for the renumbering schemes and for the original numbering of the CSM Focus Problem 1. The original numbering is a result of executing processor CSM1 to generate the model of a blade stiffened flat panel (no hole present). While this original numbering was rather simple to generate, it was an extremely poor numbering from a bandwidth standpoint. Tables 2 and 3 provide a relative comparison of IC1 among the various methods for a square and a cube respectively.

Table 1: CSM Focus Problem – Blade Stiffened Panel (391 joints)

Method	IC1	INV time‡	IC2	SSOL time‡
N/D	82515	212.0	6549	10.2
M/D	48737	121.7	5134	8.5
RCM	173388	450.5	10537	12.3
GPS	175074	448.5	10662	12.3
None	357932	881.9	15125	14.4

‡times in CPU seconds

Table 2: 20 by 20 Square Mesh		
Method	IC1	IC2
N/D	58462	6100
M/D	66041	6229
RCM	150785	10147
GPS	150785	10147
None	94469	8380

Table 3: 7 by 7 by 7 Cube		
Method	IC1	IC2
N/D	300947	12985
M/D	415403	14449
RCM	725649	20497
GPS	1220235	26137
None	468375	17101

Data Space Requirements:

In general, the Gibbs-Poole-Stockmeyer method requires the greatest amount of working space. The Reverse Cuthill-McKee and Nested Dissection methods each have the same minumum space requirement. Checks are made within RSEQ to ensure that there is enough working space available. If there is not enough room to form either the adjacency arrays or the new numbering, execution will stop and a message will be printed to the output file. The message will contain both the space required and the space available. The following table lists the exact requirements of each of the methods.

Method	Space Required
Nested Dissection	$\Omega + 3J$
Minimum Degree	$\Omega + 7J$
Reverse Cuthill-Mckee	$\Omega + 3J$
Gibbs-Poole-Stockmeyer	$\Omega + 9J + 2$

where

Ω = (Number of element types) + (Record Length) + $17 + 2(J + 1) + J * M$
J = Number of Joints
M = Maximum Connectivity

There may be problems, especially with very large models, in getting the new joint sequence through TOPO because of the space requirements of TOPO. It may be necessary to adjust the TOPO resets MAXSUB and LRAMAP upward. In some cases it may be impossible to run the new elimination sequence as the new connectivity exceeds the limit of the data space available (this seems to be true mainly with the Minimum Degree Method).

RSEQ creates the dataset JSEQ.BTAB.2.17 which may also be created manually in processor TAB with a JSEQ input list. There are several resets available which provide the user with information used and generated by RSEQ.

Available Resets:

BLIB. Library containing Element Definitions and the destination Library for the JSEQ.BTAB.2.17 dataset. Default value is 1 (and generally should be left at 1) although BLIB may be set to any integer between 1 and 20.

MAXCON. Maximum number of joints connected to any one joint. Default value is 8 which is the correct value for MAXCON for a regular, two dimensional, n by n square grid using four noded quadrilateral elements. For an n by n by n cube, using eight noded hexahedral elements, MAXCON would be 26.

MAXCON has been deliberately set small to conserve data space. While it must be set to at least the maximum connectivity of any one joint, it may be larger than this maximum. The result of resetting this parameter larger than necessary is that space will be allocated but not used. For most reasonably sized problems this will not cause any difficulties. However, for particularly large problems, this may result in allocating more space than is available.

If MAXCON is reset to a value smaller than is needed, execution will stop and the user will be asked to reset to a larger value. The output file will also contain the number of the joint at which MAXCON was *first* exceeded.

METHOD. Method of determining joint elimination sequence; Default is 0 – Nested Dissection. The four methods are:

0 – Nested Dissection
1 – Minimum Degree
2 – Reverse Cuthill-McKee
3 – Gibbs-Poole-Stockmeyer

The first two methods are fill minimizers; Reverse Cuthill-McKee is an envelope (or profile) minimizer and Gibbs-Poole-Stockmeyer is a bandwidth minimizer.

LR7. Record length set to default of 896. This reset will generally not require any changes.

LADPRT. Sets a flag for print out of adjacency information; Default value is 0 – no output of these arrays. If LADPRT is set to *anything* other than zero the full adjacency array, providing the full connectivity, in ascending order, for each node, will be printed out in the user's output file.

LJSPRT. Sets a flag for print out of the final joint elimination sequence; Default value is 0 – no print out of the joint elimination sequence. If LJSPRT is set to *anything* other than zero then the joint elimination sequence in both elimination and joint order will be printed to the user's output file.

Summary of Available Resets:

Reset	Default	Description and Permissible Values
BLIB	01	Library containing element definitions and Destination Library for JSEQ.BTAB.2.17 dataset
MAXCON	8	Maximum number of joints connected to any one joint
METHOD	0	Method of determining elimination sequence: 0 - Nested Dissection (Default) 1 - Minimum Degree 2 - Reverse Cuthill-McKee 3 - Gibbs-Poole-Stockmeyer
LR7	896	Record Length
LADPRT	0	0 - No print out of adjacency arrays to output file 1 - Print adjacency arrays to output file
LJSPRT	0	0 - No print out of joint elimination sequence to output file 1 - Print out joint elimination sequence to output file

Example of RSEQ Usage:

The example runstream included below generates a 4 by 4 square grid. The elimination sequence will be determined by the Gibbs-Poole-Stockmeyer Method and will be printed to the output file.

```
*procedure SQUARE
*def nn = 4
*def nn2= <<nn> * <nn>>
*def nel= <<nn>-1>
[xqt TAB
start <nn2>
matc
1  1.  0.1
jloc
1    0.    0.    0.    1.    0.    0.    <nn>  1  <nn>
<nn>      0.    1.    0.    1.    1.    0.
sa
1   0.1
[xqt ELD
e43
nsect=1
nmat=1
1  2  < <nn>+2 > < <nn>+1 >  1  <nel>  <nel>
[xqt RSEQ
reset METHOD=3,LJSPRT=1,LADPRT=1
[xqt DCU
toc 1
[xqt TOPO
*end
*call SQUARE
[xqt EXIT
```

Example of RSEQ Output

The following was taken directly from the log file of the Example Runstream. The adjacency array is printed as a consequence of setting LADPRT to one while the elimination sequence is printed as a result of setting LJSPRT to one.

```
EXIT ELD  CPUTIME=     1.5 I/O(DIR,BUF)=     130       2
<DM> CLOSE, Ldi:     1, File:  SQUARE.L01
METH=        3
LJSP=        1
LADP=        1
** BEGIN RSEQ **   DATA SPACE=    200000 WORDS
<DM> OPEN, Ldi:     1, File:  SQUARE.L01, Attr:  OLD, Block I/O
<DM> OPEN, Ldi:  25, File:  NS.L25 , Attr:  SCRATCH, Block I/O

ADJACENCY ARRAY:
JOINT    1 :          2    5    6
JOINT    2 :          1    3    5    6    7
JOINT    3 :          2    4    6    7    8
JOINT    4 :          3    7    8
JOINT    5 :          1    2    6    9   10
JOINT    6 :          1    2    3    5    7    9   10   11
JOINT    7 :          2    3    4    6    8   10   11   12
JOINT    8 :          3    4    7   11   12
JOINT    9 :          5    6   10   13   14
JOINT   10 :          5    6    7    9   11   13   14   15
JOINT   11 :          6    7    8   10   12   14   15   16
JOINT   12 :          7    8   11   15   16
JOINT   13 :          9   10   14
JOINT   14 :          9   10   11   13   15
JOINT   15 :         10   11   12   14   16
JOINT   16 :         11   12   15

ELIMINATION ORDER =  1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16
      JOINT ORDER =  1  2  5  6  3  7  9 10 11   4   8  12  13  14  15  16

      JOINT ORDER =  1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16
ELIMINATION ORDER =  1  2  5 10  3  4  6 11  7   8   9  12  13  14  15  16

<DM> CLOSE, Ldi:  25, File:  NS.L25
<DM> CLOSE, Ldi:     1, File:  SQUARE.L01
EXIT RSEQ CPUTIME=     1.4 I/O(DIR,BUF)=      49      13
** BEGIN TOPO **   DATA SPACE=    200000 WORDS
<DM> OPEN,  Ldi:  25, File:  NS.L25 , Attr:  SCRATCH, Block I/O
<DM> OPEN,  Ldi:  26, File:  NS.L26 , Attr:  SCRATCH, Block I/O
NO. OF   4-NODE ELEMENTS=      9
TOTAL NO. OF ELEMENTS=       9
MAXCON, MAXSUB, ILMAX=     281  39340     280
NSUBS,KSIZE,NR5,LR5=         58         20         1        896
MAXCON, MAXSUB, ILMAX=     280  39340     280
SIZE INDEX=      36,  IC1, IC2=      315       87, NR4=     1
<DM> CLOSE, Ldi:  25, File:  NS.L25
<DM> CLOSE, Ldi:  26, File:  NS.L26
EXIT TOPO CPUTIME=     2.3 I/O(DIR,BUF)=      75       7
```

APPENDIX D. Modifications to SPAR Reference Manual For NICE/SPAR Usage

THE SECTION NUMBERS BELOW REFER TO THE SECTIONS IN REFERENCE 4 TO WHICH THE MODIFICATIONS APPLY.

2.2 The Data Complex

By default, the file names corresponding to NICE/SPAR libraries are formed by appending the extension Lxx to a root file name ("NS" currently) where xx is the library number (i.e., NS.L01 for library 1).

The table of contents (TOC) is maintained by the NICE data manager in a different format than the SPAR TOC. The NICE/SPAR TOC items displayed by DCU are: sequence no., date, time, lock code, no. of records, name of creating processor, dataset name. Other items in the SPAR TOC which are required by the processors (dataset length, record length, no. of columns per block and data type) are obtained in NICE/SPAR via GAL record level utilities.

2.3 Card Input Rules

The same input rules are followed except:

1) Real data input may contain "E" at the beginning of the exponent field as in FORTRAN.

2) The comment character is # instead of $.

2.5 Dataset Structure

The SPAR dataset structure is followed except:

1) NWORDS is always an integral multiple of NI*NJ.

2) In most cases, one SPAR block corresponds to a single NICE record. However, in some datasets a SPAR block corresponds to a NICE record group, where an individual NICE record corresponds to a segment of the SPAR block.

2.5.1 Table

Tables can be of any SPAR data type; tables with ITYPE = $\pm$ 1 may not contain mixed data, but ITYPE = 0 tables may contain values of integer, real or alphanumeric type.

3.1.4 Alternate Reference Frames (ALTREF)

Material reference frames may also be defined for two-dimensional elements (see ELD SREF pointer below).

The following frames are generated automatically in TAB:

1) Global frame; x_{alt} coincident with x_{global}
2) x_{alt} coincident with y_{global}
3) x_{alt} coincident with z_{global}

While the global frame is always frame 1, frames 2 and 3 may be overwritten by the analyst. A message warning that a predefined reference frame is being overwritten will be written to output and execution will continue.

3.1.9 E21 Section Properties

Ten classes of cross section are allowed; the class RECT has been added. The required input record for RECT is:

RECT k, b1, t1

where b1 is the dimension on the 2-axis and t1 is the dimension on the 1-axis of the member reference frame.

3.2 ELD

3.2.1.2 Element Reference Frames

Change the second sentence of the first paragraph to read:

All element-related input and output (section properties, stresses, etc.) is relative to these frames *unless SREF has been set to something other than zero.*

3.2.2.2 Area Elements (E31, E32, E33, E41, E42, E43, E44)

Add to Table Pointer:

SREF	0	SREF defines the stress reference frame for the element. By default this reference frame is coincident with the element frame. The element constitutive properties are assumed to be in this frame. Stresses may be calculated in this and in the element reference frame. For SREF= n, reference frame n must have been defined via TAB/ALTREF.

Add to the bottom of the page:

Note that the mesh generators may also be used for defining networks of experimental 4-noded elements.

3.4 EKS

Add at the end of the first paragraph (Function): If required (by SREF set in ELD), EKS will perform the transformations on the element flexibility matrix.

Add to RESET Controls:

BLIB	1	Library containing element definitions, joint locations, etc.
ZK2D	.0001	Size of zero test parameter for two-dimensional elements.
GIPT	2	Number of integration points used for solid elements; may be set to 2, 3, or 4.

4.1 TOPO

Under RESET Controls, add to LRAMAP meaning:

If necessary (due to problem size) TOPO will adjust the block length to a maximum size of 4032 words. If this is still not large enough, then execution will terminate with a fatal error. This reset should then be set to at least 4032 before again trying to run the problem.

Also, add to the MAXSUB meaning:

MAXSUB = (ilmax)*(ilmax+1). When MAXSUB is reset, ILMAX is automatically re-calculated.

4.2 K - The Stiffness Matrix Assembler

Under RESET controls, the default for SPDP is 2, so double precision output is obtained because of the smaller word size on VAX.

Add to RESET Controls:

NAME	K	First word of output dataset name (e.g. default is K.SPAR, the unconstrained system stiffness matrix). The elemental arrays used in the assembly must be located in segment 5 of the EFIL dataset.

5.1 AUS

Table 5-1 Summary of AUS Subprocessors

Add the following to the list of General Arithmetic subprocessors:

LTOG
GTOL
ARAN
DRAN
MXTY
MTRA
MXV

Table 5.1.2-1 Summary of General Arithmetic Operators

Add the following:

Command Forms	Meaning
Z = ARAN(X)	Z = Rank of X in ascending order
Z = DRAN(X)	Z = Rank of X in descending order
Z = MXTY(X,Y)	Z = X Y where X and Y are multi-block datasets blocked by columns
Z = MTRA(X)	Z = X where X is a multi-block dataset blocked by columns
Z = MXV(X,Y)	Z = X Y where X is a multi-block dataset blocked by columns and Y is a single block of a dataset

5.1.2.9 LTOG and GTOL

Delete the last paragraph in the section. The paragraph begins: These commands may not be used if any joint motion components were excluded . . .

5.1.2.10 ARAN and DRAN

The form of the commands are:

Z = ARAN(X)
Z = DRAN(X)

The source dataset must be a single-block dataset of either integer or real data type. The output dataset will be a single-block dataset of integer type whose elements are the ranks (ascending or descending) of the corresponding elements of the source dataset.

5.1.2.11 MXTY, MTRA, and MXV

The source datasets must be real data type and may be multi-block. They are interpreted as rectangular matrices where each block is one column of the matrix. The destination dataset will be written one column per block.

Z = MXTY(X,Y) indicates Z = X Y
Z = MTRA(X) indicates Z = X
Z = MXV(X,Y) indicates Z = X Y

For MXTY and MXV, X and Y must be conformable for multiplication. For MXV, Y must be a single block of a dataset which may contain multiple blocks.

5.1.3.1 TABLE

The command line is:

TABLE,U(NI = ni, NJ = nj, ITYPE = n): N1 N2 n3 n4: data . . .

where the optional parameter ITYPE has been added, being the SPAR data type code of the dataset

The footnote should read:

* Loop-limit format is permitted for ITYPE = 0 or ± 1.
It is not permitted for ITYPE = 4.

5.2 DCU - The Data Complex Utility Program

The following commands are not implemented in the current version of NICE/SPAR:

XCOPY, XLOAD, REWIND, TWRITE, TREAD, NTAPE, STORE, RETRIEVE

Section 8. EIG - Sparse Matrix Eigensolver

Instead of using the Cholesky-Householder method for solving the low-order eigenproblem, the combination shift QZ algorithm described in Reference 12 is used.

10.2 PLTB

PLTB is used to produce graphical displays on Tektronix 4014 compatible terminals. The following RESET controls are provided:

Name	Default Value	Meaning
TERM	TEK	Terminal type Use TEK for Tektronix 4014; Use VT for DEC VT240
BAUD	9600	Communications baud rate
OUT	6	Unit no. for graphics output; Set to integer >6 for writing to disk file

APPENDIX E. NICE/SPAR Processor/CLIP-GAL Interface Subroutine Descriptions

Subroutine DAL (NU, IOP, KA, KORE, IEA, KADR, IERR, NWDS, NE, LB, ITYPE, NAME1, NAME2, NAME3, NAME4)

Purpose: Read or write a nominal dataset named NAME1.NAME2.NAME3.NAME4 in library NU (for single record datasets only).

Parameters:

NU	library number (integer, input)
IOP	operation code (integer, input) = -1, Rename current dataset; set KADR to dataset sequence number = 0, Set up an entry in TOC for new dataset; disable old datasets of same name; set KADR to dataset sequence number = 1, Same as 0 but also write one record of data from KA. = 2, Same as 1 but does not disable old datasets. = 10, Get TOC information without reading data; set IERR if not found. = 11, Same as 10 but also read one record (LB items) of data into KA.
KA	initial address of array containing data to be read or written; actual data type depends on ITYPE. (input for write operation, output for read operation)
KORE	number of words available for dataset (integer, input) If LB > KORE and IOP > 9, IERR set to 2. If KORE = 0, the check for space is skipped.
IEA	error condition check code (integer, input) = 1, Print message and return if error encountered. = 2, Disregard error. other, Print message and abort.
KADR	Dataset sequence number, = 0 if not found. (integer, output)
IERR	error code on return (integer, output) = 0, No error = -1, dataset not found = -2, Insufficient space for dataset
NWDS	number of items in dataset (integer, input for write, output for read)
NE	number of columns per block (integer, input for write, output for read)
LB	record size(items) (integer, input for write, output for read)

ITYPE	SPAR data type code (integer, input for write, output for read) $= 0$ integer data $= \pm 1$ real data $= \pm 2$ double precision data $= 4$ alphanumeric data
NAME1	1st component of dataset name, 4 bytes (alphanumeric, input)
NAME2	2nd component of dataset name, 4 bytes (alphanumeric, input)
NAME3	3rd component of dataset name (integer, input)
NAME4	4th component of dataset name (integer, input) (any component of the dataset name may be 4HMASK which is a "wildcard" matching parameter)

Functional description:

1. Library NU is checked to be open; if not, it is opened as a GAL82 library on disk file NS.Lxx, where xx is the library number.
2. For writing, the name is entered in TOC via GMPUNT. KADR is set to the dataset sequence number of the new dataset. If IOP = 1, one record of LB words of the appropriate type are written via GMPUTN or GMPUTC.
3. For reading, the dataset sequence number is located and the matched dataset name components are returned in common block /TOCLIN/. The dataset length, record length, rowsize, and data type are returned in /TOCLIN/ and in the argument list. If IOP = 11, the available space (KORE) is checked and one record of data is read via GMGETN or GMGETC.
4. For IOP = -1, the current dataset is renamed. No other parameters can be changed at this time in NICE/SPAR.

Subroutine FIN (NERR, NER)

Purpose: Terminate NICE/SPAR processor.

Parameters:

NERR	Error code (integer, input) $= 0$, no error $\neq 0$, 4-byte alphanumeric error code to be printed (A4)
NER	error number to be printed if NERR = 0 (I10 format) (integer, input)

Functional description:

1. Close libraries 1-20 and 27-30 conditionally; close libraries 21-26 unconditionally.
2. Print execution statistics (CPU, clock time, buffered I/O, direct I/O).
3. Print error messages according to input parameters.
4. Chain to NICE/SPAR executive via CLPUT.

Subroutine INTRO (IDPROC)

Purpose: Log processor name with data manager and get unit assignment for printed output file.

Parameter:

IDPROC - Processor name, in upper case (input, character*4)

Function description:

1. Call GMSIGN to enter processor name to be "signed" into datasets created by the processor.
2. Call ICLUNT to get the unit number assigned to the print file and assign this valve to the second integer variable in common block /IANDO/. This variable is used by NICE/SPAR processors for normal output.

Function LTOC (NU, J, NAME1, NAME2, NAME3, NAME4)

Purpose: To get item from TOC

Parameters:

NU	library number (integer, input)
J	TOC item number desired, 1-12 (integer, input)
NAME1	1st component of dataset name, 4 bytes (alphanumeric, input)
NAME2	2nd component of dataset name, 4 bytes (alphanumeric, input)
NAME3	3rd component of dataset name (integer, input)
NAME4	4th component of dataset name (integer, input)

Functional description:

1. Find dataset in NICE TOC via LMFIND.
2. Get TOC information via GMGENT, GMGETN, and LMRECS.
3. Set function value to the value of the desired item.

The TOC items are:

1	Dataset sequence number
2-4	Unused
5	Number of words in dataset.
6	Number of columns per block (for matrix type data)
7	Record size
8	SPAR data type code
9	1st component of dataset name
10	2nd component of dataset name
11	3rd component of dataset name
12	4th component of dataset name

Subroutine NSNEXT (NXTPRC)

Purpose: To store the parsed items of the next command to be picked up by the main program. This routine is furnished for processors which don't use READER to parse input and which have read (ahead) a command to execute another processor.

Parameters:

NXTPRC Name of the next processor to be executed (character, input)

Functional description:

1. If NXTPRC is non-blank, set the values in common block /INREC/* so that the main program will execute the processor named by NXTPRC immediately. Values are set as follows: NAME="STOP", IDATA(1)="[XQT", IDATA(2)=first 4 characters of NXTPRC, KIND(1)=4, KIND(2)=4, NDW=2.
2. If NXTPRC is a blank string, set the values in common block /INREC/ so that the main program will read a new command. Values are set as follows: NAME="STOP", IDATA(1)=0, KIND(1)=4, NDW=0.

* See READER for a description of the common block /INREC/.

Subroutine READ (IA, LCL, IEOF)

Purpose: Get one unparsed user input record from NICE.

Parameters:

IA	input record contents, array of LCL words, one character of input per word (integer, output)
LCL	input, number of words in array IA
IEOF	end of input flag (integer, output) = 0, successful input operation = 1, no input obtained

Functional description:

1. Initialize IA to blank.
2. Call CLGET to get an input record using the 4 character prompt string from common block /PID/. All macro expansions in the record have been performed by CLGET prior to return.
3. Store individual characters from input record into IA, one character per word, left justified.

Subroutine READER

Purpose: Get one line of user input and parse it according to the SPAR command input syntax. Input data items are stored in common block /INREC/.

Functional description:

1. Call READ to get user input record via CLGET.
2. Parse the input according to the SPAR input syntax specified in reference 4, with modifications described in Appendix D. Up to 40 items per record are allowed.
3. Return data items in common block /INREC/ described below.
4. If first item is "FIN", call subroutine FIN to terminate the processor.
5. If the first item is "RUN" or "[XQT", set NAME to "STOP". NICE/SPAR processors use this value of NAME as the end-of-input flag.

Common block /INREC/ contents:

IDATA(40)	Parsed input data items; actual data stored in IDATA may be of integer, real, or alphanumeric type.
KIND(40)	Integer SPAR data types of corresponding words in IDATA.
NAME	Alphanumeric command key; set to IDATA(1) if KIND(1) = 4 (alphanumeric); set to "STOP" if IDATA(1) = "RUN " or "[XQT"; otherwise = 0.
NDW	Number of words in the IDATA array.
NA41	Integer index in IDATA where alphanumeric label begins, if a label is included in this record.
NA42	Integer index in IDATA where alphanumeric label ends.

Subroutine RIO (NU, IWR, IOP, IDSN, KSHFT, KSHFT2, KA, L, ITYPE, NE)

Purpose: Read or write named records to NICE nominal dataset.

Parameters:

NU	Library number (integer, input)
IWR	Operation code (integer, input) = 1, Write records KSHFT:KSHFT2 = 2, Read records KSHFT:KSHFT2 = 10, Write records KSHFT:KSHFT2 and return next record number in KSHFT. = 20, Read records KSHFT:KSHFT2 and return next record number in KSHFT.
IOP	record location code (integer, input) = 1 or 2, Read or write records KSHFT:KSHFT2 = 3, Append records to end of dataset
IDSN	dataset sequence number (integer, input)
KSHFT	initial record number to be accessed (integer, input always, output if IWR>9)
KSHFT2	final record number to be accessed (integer, input)
KA	initial address of array of data (data type depends on ITYPE input for write operation, output for read operation)
L	number of data items to be read or written (integer, input)
ITYPE	SPAR data type code (integer, input)
NE	number of columns per block (integer, input for write, dummy argument for read)

Functional description:

1. Get NICE data type code from ITYPE. Construct NICE record name.
2. For IOP = 1 or 2, if IWR=1 or 10, write records via GMPUTN or GMPUTC; if IWR = 2 or 20, read record via GMGETN or GMGETC. For IOP = 3, get number of records written on dataset; construct record name DATA.nrec+1:nrec+(KSHFT2-KSHFT); write records via GMPUTN or GMPUTC.
3. For IWR > 9, return next record number in KSHFT.

Subroutine TOCO (NU, NAME, IOP, NLINE)

Purpose: Find dataset and return TOC information

Parameters:

NU	Library number (integer, input)
NAME	4 word array, dataset name (alphanumeric and integer, input)
IOP	operation code (integer, input) = 1, find first matching dataset $\neq$ 1, disable all matching datasets after entry NLINE
NLINE	dataset sequence number to start search at (integer, input, output if IOP = 1)

Functional description:

1. Find all matching datasets via GMATCH; if no matches found, set NLINE = -1 and return.
2. For IOP = 1, set NLINE = seq. no. of first matching dataset after input NLINE. Get TOC information via GMGENT, GMGETN, LMRECS. Return TOC information in common block /TOCLIN/.
3. For IOP $\neq$ 1, disable matching datasets after NLINE. Set NLINE to the number of disabled datasets.

APPENDIX F. Guidelines for Installing User Elements

The contents of this appendix are extracted from the SPAR Stuctural Analysis System Reference Manual, Vol. 4, by C. E. Jones and W. D. Whetstone, (Ref. 4) with modifications for NICE/SPAR usage.

TERMINOLOGY:

For each element, an "Element Reference Frame" is defined using the same convention as for standard SPAR element types E31, E41, S41, S61, and S81: that is, the x axis is directed from node 1 through node 2, and node 3 lies in the first quadrant of the x-y plane.

The following symbols will be used in explaining how to employ the experimental element capability:

Name	Dimension	Definition
TYPE		A 4-character alphanumeric word assigned by the user to identify a particular element formulation, analogous to the standard element type codes (E21, E43, S81, etc.). A model may consist of any of the standard element types, plus one or more types of experimental elements. There is no specific limit on the number of types of experimental elements.
MAJOR		Elements are classified as: MAJOR 1 for line elements, e.g. E21, E22. MAJOR 2 for 2D elements, e.g. E31, E43. MAJOR 3 for 3D elements, e.g. S81, F41. (Experimental elements must use MAJOR > 3)
MINOR		An integer assigned by the user to identify a specific element type. A unique "MINOR' is defined for each "type".
N		The number of nodes per element (minimum = 3, maximum = 32). Experimental beam elements should be modelled as 3-node elements, using the third point to establish the cross-section orientation.
M		N(N + 1)/2. See K, CM, KG below.
NDF		Number of degrees of freedom per node, either 3 (3 displacements), or 6 (3 displacements and 3 rotations).
NNDF		The total number of degrees of freedom per element, N times NDF. See U and F below.

Name	Dimension	Definition
X	(3,N)	Relative to the element reference frame, the direction i position coordinate of element node j is X (i,j).
U	(NNDF)	Element nodal motion vector. Where Dij and Rij are direction i displacement and rotation components (relative to the element reference frame) of element node j, the order of terms in U is as follows, if ndf=6: D11 D21 D31 R11 R21 R31 D12 D22 D32 R12 - - If NDF = 3, the order is the same, except the rotation terms are not present. None of the user-written subroutines access either U or F, defined below. U and F are defined here only for use in defining other arrays which must be generated in user-written routines.
F	(NNDF)	Element nodal force vector, corresponding to U.
P N4PROP	(NP)	Element property array. The content of P is established by the user. Typical items are material and section properties, option controllers, etc. Before executing ELD, the user must create, via AUS/TABLE, a table named XXXX BTAB 2 N4PROP, where N4PROP is any integer greater than 100. Each line in this table is a P array, applicable individual elements, as established via the NSECT to one or more pointer in ELD.
S	(NS)	The element "stress" state vector. The content of S is established by the user. Typical terms in S are stress field coefficients, strains, stress resultants, etc. See the definition of Q, R, and D below.
Q	(NQ)	The element "thermal" load vector. The content of Q is established by the user. Typical components of Q are coefficients of temperature or dislocation functions, temperature gradients, etc., defined in any manner the user finds convenient. See the definition of the C and D matrices below. The Q vectors for individual elements are constructed via AUS/ELDATA. The NODAL TEMPerature and NODAL PRESsure arrays, if present, do not apply to experimental elements.
Y	(N)	Element nodal weight distribution. The weight of an element = Y(1) + Y(2) + - - - Y(n). Processor E uses Y in constructing the system diagonal mass matrix, DEM.

Name	Dimension	Definition
K	(NDF,NDF,M)	The element stiffness matrix, arrayed as submatrices, each dimensioned NDF by NDF: $$K = \begin{bmatrix} K(-,-,1) & K(-,-,2) & K(-,-,4) & - & - & - \\ & K(-,-,3) & K(-,-,5) & - & - & - \\ & & K(-,-,6) & - & - & - \\ \text{Symmetric} & & & - & - & - \\ & & & K(-,-,M) & & \end{bmatrix}$$
C	(NNDF,NQ)	Thermal force influence matrix. Total element nodal forces are: $F = K\ U + C\ Q$
R	(NS,NNDF)	Mechanical stress recovery matrix.
D	(NS,NQ)	Thermal stress recovery matrix. The total stress, as computed by GSF, is: $S = R\ U + D\ Q$
CM	(NDF,NDF,M)	Element consistent mass matrix, arrayed the same as K.
KG	(NDF,NDF,M)	Element initial-stress stiffness matrix, arrayed the same as K.
T(3,3)		Transformation matrix for element-to-global coordinate transformation.
SPDP		An integer code specifying the precision of the element stiffness matrix (user input in ELD) 1 = single, 2 = double
SREF		An integer code specifying the element stress reference frame (user input in ELD) 0 = material x-axis and element x-axis coincide 1 = material x-axis and global x-axis coincide 2 = material y-axis and global x-axis coincide 3 = material z-axis and global x-axis coincide

USER WRITTEN SUBROUTINES:

To implement experimental elements, the user must code the following routines and incorporate them into the indicated processors, replacing empty routines having the same names in the standard version of SPAR. It is permissible to omit DMEXPE, CMEXPE, and KGEXPE.

Processor	Routine Name	Argument List	
		Inputs:	Outputs:
E	DMEXPE(	MAJOR,MINOR,N,NDF,P,X,	Y)
EKS	KEXPE(	MAJOR,MINOR,N,NDF,P,X,T,SREF,	K,R,C,D)
M	CMEXPE(	MAJOR,MINOR,N,NDF,P,X,	CM)
KG	KGEXPE(	MAJOR,MINOR,N,NDF,P,X,S,	KG)

The primary limitation on the experimental element capability is the total length of an element information packet in the E-state, which is initiated in processor E. The component parts are:

Part	Length	Contents
1	12 + N	
2	0	
3	18 + 27N	X, T, other element transformation data
4	NP	P
5	M × NDF × NDF × SPDP	K
6	NS × NNDF	R
7	NS	S
8	NNDF × NQ	C
9	NS × NQ	D

The related limitations are:

- The sum of the lengths of parts 1 through 4 may not exceed 1000.
- All processors accessing the E-state (e.g., K, M, KG, EQNF, GSF) require sufficient central memory to load at least one complete element information packet, in addition to the other c.m. requirements, as defined in the SPAR Reference Manual.
- In E, the entire XXXX BTAB 2 N4PROP array must be in central memory.

Although the experimental element capability has been structured such that there are few size restrictions, users should be aware that extremely large element information packets will generally result in relatively high I/O costs.

PROGRAM EXECUTION:

Executions proceed the same as in a standard analysis, except that before executing ELD, the user must execute AUS/TABLE to construct, for each type of experimental element, a table named XXXX BTAB 2 N4PROP. The identifying integer, N4PROP, may be any number greater than 100. Each line in this table is a P array, as previously defined, containing the material and section constants applicable to one or more elements.

It is also mandatory that a MATERIAL CONSTANT dataset be generated in TAB, even though the experimental elements do not reference MATC. This is required because processor E automatically loads MATC, and retains it in central memory throughout execution.

Experimental elements are defined in ELD as illustrated below. Within the same ELD execution, other element types, either standard or experimental, may also be defined.

```
[XQT ELD
#
# Define all elements of a given type:
#
EXPE TYPE, MAJOR, MINOR, N, NDF, NS, NQ, N4PROP, SPDP
SREF=i
NSECT=k      # Line K of XXXX BTAB 2 N4PROP is ''P'' for
#  elements defined subsequently.
j1 j2 - - jn#  First element connects joints j1 j2 - -jn.
j1 j2 - - jn#  Second element
-
-
NSECT=j # Line j of XXXX BTAB 2 N4PROP is ''P'' for elements defined
#   subsequently.
j1 j2 - - jn#
j1 j2 - - jn#
```

NSECT is the only table pointer command applicable to experimental elements. The MOD, INC, and GROUP commands function the same as for the standard elements. The standard mesh generation facilities may be used for experimental elements having the same nodal arrangement as standard elements.

The SREF command applies to experimental elements to designate the stress reference frame for subsequently defined elements. The default value assumed for SREF is zero; the current established value stays in effect until superseded by another SREF command. Upon conclusion of execution of the EXPE subprocessor, the SREF value reverts to the default value of zero.

If thermal loading is present, an ELDATA-format dataset named TEMP type iset icase must be created via AUS/ELDATA before EQNF is executed:

```
[XQT AUS
ELDATA:  TEMP type iset icase
G=1
e=1:     Q1 Q2 - - Qnq#  Q(nq items) for element 1, group 1.
e=2:     Q1 Q2 - - Qnq#  Q(nq items) for element 2, group 1.
-
-
-
```

The functions performed by SPAR processors in support of experimental elements are summarized below. None of the other SPAR processors (e.g. the plot programs) recognize experimental elements.

Processor	Function
ELD	Processes element definitions, and creates DEF, GD, GTIT, and DIR datasets.
TOPO	Accounts for presence of experimental elements in constructing the topological arrays, KMAP, AMAP, enabling system matrix assembly and factoring to be performed.
E	Creates the E-state datasets for experimental elements, in skeletal form. Adds contribution of experimental element weights to DEM.
EKS	Inserts the K, R, C, and D arrays into the E-state.
K	Includes experimental element K's in system K.
M	Includes experimental element CM's in system CEM.
KG	Includes experimental element KG's in system KG.
AUS	Permits loading of TEMP type datasets via ELDATA.
EQNF	Adds contribution of experimental element fixed-joint forces to equivalent nodal force arrays.
GSF	Computes S for each element, including thermal effects, present. S is embedded in the E-state, for use by KG, if requested via the EMBED reset control. S is not printed by GSF unless ONLINE=2. STRS Type datasets are not produced by GSF for experimental elements. PSF does not recognize experimental elements.

References

1. Felippa, Carlos A.: Architecture of a Distributed Analysis Network for Computational Mechanics. *Computers and Structures*, vol. 13, 1981, pp. 405-413.

2. Felippa, Carlos A.: *A Command Language for Applied Mechanics Processors, vols. 1-3.* LMSC-D 78511, November 1983.

3. Felippa, Carlos A.: *The Global Database Manager EZ-GAL.* LMSC-D766995, November 1982.

4. Whetstone, W. D.: *SPAR Structural Analysis System Reference Manual, vols. 1-4.* NASA CR 158970-1, December 1978.

5. Whetstone, W. D.: Computer Analysis of Large Linear Frames. *J. Struct. Div.*, ASCE, vol. 95, no. ST11, Proc. Paper 6897, November 1969, pp. 2401-2417.

6. Marlowe, M. B.; Moore, R. A.; and Whetstone, W. D.: *SPAR Thermal Analysis Processors Reference Manual, System Level 16.* NASA CR 159162, October 1979.

7. Pian, T. H. H.: Derivation of Element Stiffness Matrices by Assumed Stress Distributions. *AIAA J.*, vol. 2, 1333-1336, 1964.

8. Cunningham, Sally: *SPAR Data Set Contents.* NASA TM 83181, October 1981.

9. G. M. Stanley, et al: *Preliminary Development of a Testbed for Computational Structural Mechanics, Part 1.* LMSC-D067201, June 1986.

10. George, Alan and W-H Liu, Joseph: *Computer Solution of Large Sparse Positive Definite Systems.* Prentice-Hall, Inc., Englewood Cliffs, NJ, 1981.

11. Everstine, G. C: *The Bandit Computer Program for the Reduction of Matrix Bandwidth for NASTRAN.* NSRDC Report 3872, March 1972.

12. Ward, R. C.: *An Extension of the QZ Algorithm for Solving Generalized Matrix Eigenvalue Problems.* NASA TND-7305, July 1973.

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TM-89096		

4. Title and Subtitle	5. Report Date
Introduction to the Computational Structural Mechanics Testbed	September 1987
	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
C.G.Lotts; W.H.Greene; S.L.McCleary; N.F.Knight, Jr.; S.S.Paulson; and R.E.Gillian	

9. Performing Organization Name and Address	10. Work Unit No.
NASA Langley Research Center Hampton, VA 23665-5225	505-63-11-07
	11. Contract or Grant No.

12. Sponsoring Agency Name and Address	13. Type of Report and Period Covered
National Aeronautics and Space Administration Washington, DC 20546-0001	Technical Memorandum
	14. Sponsoring Agency Code

15. Supplementary Notes

C.G.Lotts, S.L.McCleary, S.S.Paulson: PRC/Kentron, Inc.

16. Abstract

The CSM Testbed software system based on the SPAR finite element code and the NICE system is described. This software is denoted NICE/SPAR. NICE was developed at Lockheed Palo Alto Research Laboratory and contains data management utilities, a command language interpreter, and a command language definition for integrating engineering computational modules. SPAR is a system of programs used for finite element structural analysis developed for NASA by Lockheed and Engineering Information Systems, Inc. It includes many complementary structural analysis, thermal analysis, and utility functions which communicate through a common database. The work on NICE/SPAR was motivated by requirements for a highly modular and flexible structural analysis system to use as a tool in carrying out research in computational methods and exploring new computer hardware. Analysis examples are presented which demonstrate the benefits gained from a combination of the NICE command language with a SPAR computational modules.

17. Key Words (Suggested by Authors(s))	18. Distribution Statement
Structural analysis software Finite element analysis Finite element software Thermal analysis software	Unclassified—Unlimited Subject Category 39

19. Security Classif.(of this report)	20. Security Classif.(of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	173	A08

NASA FORM 1626 OCT 86

For sale by the National Technical Information Service, Springfield, Virginia 22161-2171